ANTEPRIMA: IBM DB2 VIPER
IL DATABASE DI BIG BLUE CON SUPPORTO NATIVO A XML, MA ANCHE TANTO ALTRO...

Rivista + "Le grandi guide di ioProgrammo" n°7 a € 12,90 in più

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90 | VERSIONE STANDARD RIVISTA+CD €6,90

PER ESPERTI E PRINCIPIANTI | Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • NOVEMBRE 2006 • ANNO X, N.11 (108)

REALTÀ VIRTUALE
SUBITO NEI TUOI SOFTWARE

Muovi finestre e icone a distanza, usando una piccola sorgente luminosa e una webcam. In pieno stile "MINORITY REPORT"

Scopri le librerie Grafiche Intel OpenSource Computer Vision Library

Acquisisci il punto luce con la telecamera e isolala dai disturbi dello sfondo

Aggancia i movimenti della fonte luminosa a quelli del mouse e fallo muovere senza toccarlo

RITORNO AL TURBO PER BORLAND

IN PIÙ LA VIDEOGUIDA NEL CD

Scopri tutte le novità del nuovo compilatore C# e realizza un'applicazione completa in pochi minuti usando ECO

JAVA

CREA IL TUO BROKER DI BORSA
Scarica i dati da internet, elaborali ed il tuo consulente personale è pronto!

PRENDI I DATI DALLE FOTO
Utilizza EXIF per estrarre informazioni come la data di scatto e l'esposizione

ECLIPSE IN 7 PASSI
Inizia subito a programmare con l'ambiente che ti rende più semplice la vita

NOVITÀ

IIS 7.0 UN CERBERO PER LA SICUREZZA
Ecco i nuovi moduli del Web server di MS. Scopri come rendono il tuo sito inattaccabile

ALGORITMI

DIVIDERE BENE LE RISORSE
Avete 5 scialuppe di salvataggio e 50 persone, come li distribuite affinché il peso sia omogeneo?

STRUMENTI

REVISIONI SOTTO CONTROLLO
Impara a usare SubVersion il software che tiene traccia delle modifiche che fai al tuo codice

ASP.NET

DOTNETNUKE IL PORTALE "PRECOTTO"
Ecco come installare un CMS in pochi minuti ed estenderlo con moduli

.NET FRAMEWORK

DISTRIBUIRE APPLICAZIONI OFFICE
Facciamolo con ClickOnce e ogni eventuale aggiornamento sarà automatico!

SPECIALE WEB 2.0
TRE ESEMPI COMPLETI PER LAVORARE SUBITO CON LE TECNOLOGIE CHE STANNO CAMBIANDO INTERNET

OPENLASZLO Crea applicazioni Flash. É semplice ed è gratuito!

AJAX Per creare pagine Web che non fanno refresh

JSON Quasi come XML ma più facile da usare

SOLUZIONI Algoritmi evolutivi, ovvero sviluppare software che autoapprende dalle proprie esperienze

Anno X - N.ro 11 (108) - Novembre 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: C. Bellucci, L. Buono, D. De Michelis, F. Grimaldi, M. Scala, A. Pelleriti, M. Locuratolo, L. Corias

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90 Offerte valide fino al 30/11/06 costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Servizio Abbonati:
☎ tel. **02 831212**
@ e-mail: **servizioabbonati@edmaster.it**

Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Ottobre 2006

Certificato n.5670 del 16/12/2005
ITportal
L'Universo Tecnologico
www.itportal.it

TEMPO DI RIPRESA

Così in Europa si prospettano anni migliori per l'economia. Non che in precedenza la consueta frase "C'è una timida ripresa" non fosse di casa negli ambienti dei mercati che contano, tuttavia questa volta sembra essere pronunciata con maggiore convinzione. Da un'eventuale ripresa dell'economia proprio noi programmatori potremmo trarne enormi vantaggi in fatto di commesse. Di fatto è sempre il settore tecnologico quello che per primo viene aggiornato quando c'è la possibilità di effettuare un investimento. Il segnale di quanto questo possa essere vero ci potrebbe essere fornito dall'imminente SMAU. Al momento in cui scriviamo non ci è dato sapere quali siano stati i temi portanti della più grande manifestazione nazionale sull"ICT e quali di questi temi abbiano incontrato i favori del pubblico. Tuttavia è facile interpretare come un segnale di fiducia un eventuale aumento di numero degli stand dei prodotti legati allo sviluppo. Viceversa potremmo sospettare che anche questa volta la "timida ripresa" non è che una frase di circostanza. Non che SMAU sia sempre un indicatore affidabile dell'economia, tuttavia certamente se ne può trarre qualche pur contestabile indicazione. Per conto nostro ipotizziamo fin da adesso una presenza massiccia del mobile. Anche se in verità alle tante comunicazioni sbandierate dai media in merito al dilagare di dispositivi mobili non ha fatto riscontro un aumento delle commesse in tal senso verso noi programmatori. E' probabile perciò che pur risultando veritieri i grafici di vendita di cellulari e palmari, è anche vero che non si pretende da questi oggetti un uso che li sostituisca ai dispositivi da tavolo. Per il resto immaginiamo qualche innovazione nel campo del Voice Over IP, del GPS e della domotica. Dove dovremo dirigerci noi programmatori? La risposta è sempre la stessa: ovunque ci sia una richiesta. Questo significa che un eventuale ripresa per noi dovrebbe essere concretizzata, in termini di investimento, nell'aggiornamento. In un mondo che si muove alla velocità di Internet è impensabile rimanere legati per troppo tempo ad un linguaggio o ad una tecnologia, mentre invece è importante essere in grado di soddisfare le esigenze sempre più complesse di un mercato che si muove nel segno dell'interoperabilità-.

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

REALTÀ VIRTUALE
SUBITO NEI TUOI SOFTWARE

Muovi finestre e icone a distanza, usando una piccola sorgente luminosa e una webcam. In pieno stile "Minority" Report

- ✔ Scopri le librerie Grafiche Intel OpenSource Computer Vision Library
- ✔ Acquisisci il punto luce con la telecamera e isolala dai disturbi dello sfondo
- ✔ Aggancia i movimenti della fonte luminosa a quelli del mouse e muovilo a distanza

pag. 14

BORLAND RITORNO CON IL TURBO!!!

Scopri tutte le novità del nuovo compilatore C# e realizza un'applicazione completa in pochi minuti usando ECO

IOPROGRAMMO WEB

SISTEMA

NETWORKING

SISTEMA

SISTEMA

GRAFICA

RUBRICHE

ANTEPRIMA

NETWORKING

CORSI BASE

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

RIVISTA + CD-ROM in edicola

BORLAND TURBO C#

Il compilatore che torna!

Borland ha segnato profondamente nel corso del tempo l'evolversi della programmazione. Già agli inizi degli anni 90, quando ancora buona parte del mondo programmava in modo procedurale, Borland era arrivata sul mercato con l'Object Turbo Pascal, primo esempio di linguaggio interamente ad oggetti dedicato alla produttività individuale. Immediatamente dopo aveva fatto segnare l'ennesima rivoluzione del mercato della programmazione lanciando la moda degli IDE Rad, era la volta di Delphi. Nel corso del tempo, tuttavia aveva cambiato radicalmente strategia e si era dedicata ai prodotti ALM tralasciando quello che era il suo mercato storico di riferimento, ovvero quello degli IDE e compilatori dedicati alla produttività individuale. Oggi questa situazione è radicalmente cambiata. Borland ha appena fondato una società apposita per riprendere lo sviluppo dei suoi IDE...

Prodotti del mese

OpenLaszlo 3.3.3

Quasi come Flex ma OpenSource

L'idea è semplice. C'è un file XML, l'utente richiama questo file per mezzo del browser. Il compilatore sul server lo compila on the fly e restituisce all'utente una pagina flash. La tecnica è interessante perché consente di creare applicazioni WEB 2.0 senza utilizzare AJAX e producendo sempre output adeguati e compatibili con ogni browser. In realtà OpneLaszlo non è l'unico prodotto che si comporta in questo modo, ma sicuramente è l'unico prodotto non commerciale a consentire questo genere di tecnica. L'alternativa commerciale è Macromedia Flex 2.0. Le differenze a livello tecnologico non sono eccessive. Tuttavia OpenLaszlo appare ancora leggermente più macchinoso rispetto al più blasonato rivale. In tutti e due i casi lo scopo è comunque realizzare web application che si comportino come applicazioni standalone

[pag.108]

DotNetNuke 4.3.5 Starter Kit

Il portale "Precotto" targato .NET

Un CMS completo e dotato di funzionalità realmente avanzate. In questo numero presentiamo uno "Starter Kit" ovvero un template per l'ambiente Visual Studio, che consente di installare il prodotto ma anche di estenderlo per coloro che avessero bisogno di funzionalità particolari. In questo numero presentiamo un bell'articolo di Carmelo Scuderi che ci porta passo passo alla realizzazione di un completo Cross Engine. DotNetNuke è particolarmente completo sia nel numero di funzionalità esposte, sia nell'organizzazione del codice. Se in molti si possono accontentare dell'installazione classica, sicuramente i più esperti approfitteranno della disponibilità del codice sorgente per modificarne il comportamento

[pag.108]

Svn server 1.4.0

Per tenere sotto controllo le modifiche

Quante volte avete eseguito una modifica al vostro codice e poi avete desiderato tornare indietro? Ecco che in questi casi un software per il controllo di versione è indispensabile. Consente di tenere traccia con dei log di tutte le modifiche apportate al vostro progetto ed eventualmente di tornare indietro ad una versione precedente oppure produrre un log degli aggiornamenti.
Non erano in molti a scommettere sul successo di questo software quando qualcuno ha cominciato a proporlo come alternativa all'onnipresente CVS. Tuttavia la buona stabilità e le ottime prestazioni nel tempo gli hanno consentito di scalare molte posizioni. Persino Linus Torvads da sempre devoto a CVS ha deciso qualche anno fa di spostare lo sviluppo del kernel su un sistems subversion. Una bella soddisfazione per Tigris

[pag.108]

Apache Tomcat 5.5.17

Programma in Java per il Web

Chi vuole programmare in JSP non può esimersi dall'utilizzare questo Application Server. Leggero ma completo si tratta di uno dei server più diffusi per JSP. In questo numero di ioProgrammo lo utilizziamo in congiunzione ad OpenLaszlo. Tomcat è un completo Application server per applicazioni JSP. Nel tempo sta lentamente subendo la concorrenza di sistemi quali JBoss, tuttavia rimane una delle alterna ve più indicate sia per iil testing di applicazioni in locale sia per la reale messa in produzione. Tomcat è facilmente installabile sia in sistemi Linux sia in sistemi Windows, dispone di un'interfaccia di amministrazione sufficientemente amichevole e si è dimostrato sempre efficiente e stabile. A tutto ciò bisogna aggiungere che è un software OpenSource

[pag.108]

RIVISTA + LIBRO + CD-ROM in edicola

I contenuti del libro

Comprendere XML

Come sempre le tecnologie di maggior successo nascono per risolvere un problema reale. XML è nato come linguaggio universale per l'interscambio dei dati. Tuttavia la sua flessibilità è risultata talmente elevata da garantirgli immediatamente un impiego in quasi tutti i campi dello sviluppo. Sia che si parli di Web Services, sia che si parli di normali servizi sul web come gli RSS, ormai XSL fa da piattaforma ad una gamma incredibilmente vasta di applicazioni. Persino i server di database più evoluti lo hanno ormai adottato come formato standard per la persistenza dei dati. In questo libro, scritto sapientemente da Francesco Smelzo, si introducono i principi base che ne consentono un utilizzo efficace. Acquisire una buona padronanza di XML vi consentirà di riflesso di comprendere a fondo la moderna programmazione.

IL LINGUAGGIO "PIATTAFORMA" PER UTILIZZARE A FONDO TUTTE LE TECNICHE DELLA PROGRAMMAZIONE MODERNA

- **Introduzione a XML**
- **La sintassi e gli schemi**
- **Il Document Object Model**
- **Lavorare con XPATH**

News

ENNESIMO BUG PER INTERNET EXPLORER

Questa volta si tratta di VML, ovvero quella parte del browser che implementa il rendering del Vector Markup Language. Riferita in questi termini potrebbe sembrare una falla non grave, ed invece è già diventata il centro di raccolta per hacker di vario genere. Di fatto sfruttando questo bug è possibile ottenere il pieno controllo del sistema vittima. Al momento in cui scriviamo Microsoft non ha ancora rilasciato alcuna patch per questo problema, viceversa esiste una patch non ufficiale creata dallo dal Zero-day Emergency Response Team e disponibile presso l'indirizzo: *http://isotf.org/zert/download.htm.* Trattandosi di una patch non ufficiale non gode ovviamente di alcun tipo di supporto

UN CONCORSO PER GURU DELL'ICT

Avete inventato un servizio innovativo? State lavorando ad un progetto a cui nessuno mai pensato? sognate di diventare il programmatore che cambierà totalmente la faccia dell'informatica? è il momento di partecipare alle selezioni per Lo European ICT Prize 2007. La Deadline per la partecipazione è fissata per il 4 Dicembre 2006. La manifestazione si avvale della partecipazione della Commissione Europea e prevede un premio di 5000 euro da destinarsi ai 20 migliori progetti che parteciperanno al contest. Il migliore di questi vincerà un premio di 200.000 Euro. I vari progetti saranno sottoposti alla volutazione di una selezione di esperti provenienti da 16 nazioni europee, i premi infine saranno assegnati da una commissione scientifica Europea. La manifestazione si prefigge di avere un forte legame con il mercato ma anche un elevato contenuto tecnico. I progetti vincitori dovranno avere caratteristiche innovative ma dovranno essere anche valutati come in grado di essere apprezzati a livello commerciale. La presenza di una commisione no profit è garanzia di trasparenza nell'assegnazione dei premi

DISPONIBILI GLI MS EXPRESSION TOOLS

Nella spasmodica ricerca di produrre strumenti che favoriscano la creatività ed aumentino contemporaneamente la produttività, ecco che Microsoft lancia i suoi Expression Tools. Si tratta di una gamma di tre prodotti, rispettivamente: Graphic Designer, Interactive Designer, Web Designer. I tre nuovi pacchetti vanno ad occupare la fascia intermedia compresa fra il livello di sviluppatore e quello di designer. In realtà potrebbero essere utilizzati con profitto sia da un programmatore che abbia bisogno durante la pratica quotidiana di produrre elementi di grafica sia da un grafico che volesse introdurre elementi di dinamicità all'interno delle proprie pagine web per esempio.

Graphic designer è

AJAX CERCA UNO STANDARD

Senza dubbio Ajax è uno dei temi scottanti del momento, grazie a questa tecnologia che consente di realizzare Web Application che non necessitano di reload della pagina, i software che girano su piattaforma Internet si stanno sempre più avvicinando a quelli standalone. Il risultato è quello di ottenere una maggiore usabilità e quello di poter usufruire all'interno di applicazioni web di caratteristiche che sono normalmente in dotazione solo alle applicazioni standalone, portandoci a parlare di Web 2.0. Nonostante che Ajax sia tutto sommato una tecnologia semplice basata essenzialmente su Javascript, è proprio questa sua semplicità a preoccupare i maggiori produttori di software dedicati allo sviluppo. Infatti i vari oggetti che ormai popolano i vari framework e che servono per gestire il Markup HTML nascondendone la complessità al programmatore, mal comunicano fra loro, con il risultato che si ha un proliferare di tecniche che presentano una sintassi e una semantica diverse fra loro e che rendono particolarmente disomogeneo un paradigma come Ajax, che invece per sua natura è decisamente standard. E' per questo che OpenAjax Alliance ha dato vita al progetto HUB, che riunisce intorno a se big del calibro di IBM, Oracle e Sun, e che mira a mettere ordine nel mondo Ajax. Certe volte ci stupiamo di come sia sorprendentemente semplice rendere complesse tecniche anche molto lineari come Ajax. Tuttavia uno standard è ormai una necessità

un prodotto specificatamente indirizzato ai grafici, consente di realizzare forme vettoriali ma anche di manipolare immagini Bitmap. Inoltre ovviamente è dotato di tutta quella serie di effetti e filtri che consentono di realizzare elementi grafici particolarmente accattivanti. Interactive Designer viceversa comincia a spostare il suo target applicativo verso i programmatori. Il suo scopo è quello di fornire strumenti adatti alla progettazione di interfacce grafiche. In questo senso incorpora anche caratteristiche di modellatore 3D. WEB Designer infine si sposta decisamente verso il lato del programmatore, proponendosi come strumento di programmazione per Web Applications. I nuovi prodotti targati Microsoft vanno a riempire quel vuoto costituito dagli ambienti per la "produttività uindividuale" fino ad ora troppo snobbati a favore di grandi realtà industriali. Tuttavia i piccoli programmatori hanno ancora un peso economico notevole sul mercato della programmazione

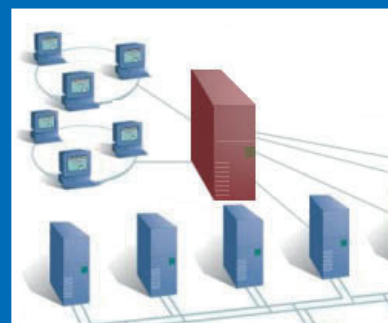
ARRIVA NETBEANS FOR BEGINNER

NetBeans è il popolare IDE per Java sponsorizzato da Sun Microsystem. Si caratterizza come ambiente spiccatamente RAD ed è probabilmente l'unico rivale dell'ormai onnipresente Eclipse. Recente la comunità che presiede allo sviluppo di Netbeans insieme a Sun, ha annunciato la disponibilità di un IDE "Ristretto" per favorire l'apprendimento del linguaggio a chi si avvicina alle tecnologie Java. La nuova versione prenderà il nome di BlueJ Edition.. Il nome deriva da un progetto nato inizialmente in Australia proprio per favorire l'apprendimento del linguaggio a coloro che vi si avvicinavano per la prima volta. BlueJ nella sua versione originale includeva anche elementi fortemente didattici pensati appositamente per guidare i nuovi arrivati alle tecniche di programmazione sottostanti i linguaggio di Sun. Netbeans BlueJ integrerà il meglio del conosciutissimo RAD con gli elementi fortemente didattici di BlueJ, il tutto dovrebbe portare alla produzione di un IDE particolarmente utile a chi si appresta ad apprendere queste tecniche di programmazione

NUOVI PRODOTTI PER INTEL

Intel, già produttrice di compilatori e framework particolarmente votati alla creazione di applicazioni fortemente ottimizzate per la propria piattaforma hardware, ha appena compiuto un ulteriore passo avanti verso la programmazione MutiThreading. In realtà si tratta di un aggiornamento globale della famiglia di prodotti coinvolti nella piattaforma MultiThreading, in particolare l Threading Building Blocks 1.0 per Windows, Linux e Mac OS X, il Thread Checker 3.0 per Windows e Linux e il Thread Profiler 3.0 per Windows. I prodotti in questione formano un circolo che chiude l'intera gamma delle problematiche relative a questo genere di programmazione. Si parte dalle librerie in C++ per la realizzazione di codice performante, fino all'uso dei profiler che consentono di controllare dove un software ha una diminuizione delle prestazioni dovuta ad un collo di bottiglia del codice compilato o del flusso dell'applicazione stessa. I vari tool possono essere utilizzato in maniera standalone, ma si possono anche integrare con Microsoft Visual Studio 2005

NASCE HADOOP PER IL CLUSTERING

Un framework votato alla creazione di applicazioni Java che possano essere eseguite in parallelo su una serie di calcolatori. Si tratta di un esperimento interessante che coinvolge problemi di particolare rilevanza. L'idea è che le informazioni possano risiedere in un qualunque momento in una qualunque parte del cluster e che l'applicazione di base possa essere suddivisa in particelle elementari eseguibili singolarmente, ognuna secondo la propria posizione nel flusso del programma, da una qualunque macchina partecipante al cluster. La tecnologia sottostante è quella del map/reduce che tramite sofisticati algoritmi implementa appunto la gestione della suddivisione particellare dell'applicazione. Hadoop non trascura il file system, che viene gestito in maniera trasparente suddividendo i dati nelle varie parti del cluster, ma lasciandone all'utente una visualizzazione monolitica. L'intero progetto deriva da Lucene, il noto framework opensource per l'indicizzazione dei contenuti, in pieno stile Google Desktop

UN MOUSE IN STILE MINORITY REPORT

IN MOLTI SI RICORDERANNO IL MITICO FILM IN CUI TOM CRUISE IN UNO SCENARIO SURREALE MUOVEVA CARTELLE E IMMAGINI SULLO SCHERMO UTILIZZANDO LE MANI COME UNA SORTA DI MOUSE VIRTUALE. QUESTO È ESATTAMENTE QUELLO CHE FAREMO...

In questo articolo tenteremo un'operazione utile quanto interessante. Cercheremo di muovere il mouse attraverso l'uso di una lampadina. Ovviamente implementeremo anche le funzioni del click e le altre funzioni tipiche del mouse. L'applicazione assomiglierà molto a quella futuristica proposta in Minority Report da Tom Cruise, dove gli oggetti sullo schermo venivano mossi sulla base del movimento delle mani. Sarebbe infatti facile indossare un guanto con una fonte luminosa innestata, per far fare al nostro mouse quello che vogliamo… anche a distanza. Dal punto di vista strettamente tecnico quello di cui abbiamo bisogno è una webcam anche di bassa qualità e una lampadina.
La Webcam si occuperà di intercettare il segnale luminoso e convertire il suo movimento da informazioni comprensibili dal puntatore del mouse.

Fig. 1: **Fotogramma originale estratto dal flusso video generato dalla webcam**

INDIVIDUARE UNA LUCE

Per riuscire nell'intento preposto, bisogna affrontare un problema concettualmente molto difficile che è quello di riuscire ad individuare una sorgente luminosa all'interno della scena ripresa dalla videocamera: per riuscirci possiamo ricorrere ad un semplicissimo trucco, nato da una osservazione di natura sperimentale. Prendiamo un singolo fotogramma di una ripresa: la sorgente luminosa, nel nostro caso un lampadina a led accesa, viene vista all'interno dell'immagine come un'area il cui colore tende al bianco. Per isolarla dal resto, cioè per "cancellare" lo sfondo che non ci interessa, sarebbe sufficiente quindi ricercare in ciascun fotogramma della ripresa tutte le aree il cui colore è proprio bianco (in RGB sarebbe 255,255,255). Sarebbe bello se tutto fosse così semplice, tuttavia molto del risultato è influenzato dal tipo di webcam usata, o meglio dal device di acquisizione video collegato; l'uso di una webcam a basso prezzo, come quella con cui abbiamo realizzato la **Figura 1**, tende a modificare pesantemente i colori reali tanto che la sorgente è vista azzurrina. Usando dispositivi di acquisizione più sofisticati, la qualità della ripresa aumenta considerevolmente anche perché intervengono strumenti hardware per la compensazione dei colori e della luminosità. Non potendo sapere a priori quale sarà lo strumento utilizzato, possiamo rendere il discorso più generale andando a cercare nel singolo fotogramma non più le aree di colore bianco ma quelle che superano un valore di soglia variabile. Per semplificare ancora di più, per evitare di dover regolare un modello automatico di elaborazione basato su tre parametri (R, G, B), possiamo convertire l'immagine in toni di grigio (R=G=B) in modo da avere un singolo valore da controllare, variabile tra 0 (nero) e 255 (bianco). Mediamente un buon risultato può essere ottenuto con un valore di soglia intorno a 240 per una webcam a basso costo. Naturalmente, più è alta la qualità del dispositivo più questo parametro tenderà ad avvicinarsi al valore massimo 255. Una volta isolata la "macchia", cioè l'area sul frame occupata dalla sorgente luminosa, si può operare un'ulteriore semplificazione cercando di rappresentarla come un singolo punto da cui si irradia la luce. Possiamo determinare il centro di questa area e legare le coordinate di questo punto sull'immagine con le coordinate del puntatore del mouse sullo schermo del computer, metà del gioco è fatto...

GLI STRUMENTI

Alla luce di questa osservazione possiamo mettere in atto una serie di semplificazioni che riducono di molto la complessità del problema.
In sostanza quello che bisogna realizzare è un sistema che svolga i passi mostrati in **Figura 2**.
Per implementare i vari algoritmi di elaborazione e di valutazione dei singoli frame estratti dal flusso video, utilizzeremo una libreria sviluppata da Intel Corporation sotto licenza OpenSource. Questa libreria, la OpenCV, dispone di una serie di strumenti altamente specializzati nel campo dell'interpretazione delle immagini: determinare un movimento di oggetti, interpretare il linguaggio dei segni o il movimento delle labbra e tanto altro. L'utilità è indiscutibilmente legata alla realizzazione di applicazioni di ausilio per i disabili o per migliorare l'interazione uomo-macchina, come nel nostro caso in quanto cerchiamo di realizzare un'alternativa al classico mouse basato sull'individuazione di una sorgente luminosa. Il pacchetto distribuito dalla Intel contiene tutto il codice sorgente scritto in C ed è privo di royality e quindi liberamente distribuibile.

INDIVIDUAZIONE DEL DEVICE

Per interfacciarsi con il dispositivo di acquisizione video è possibile utilizzare diversi strumenti software. Anche la libreria OpenCV mette a disposizione delle utility per individuare e connettersi ai device, ma fa uso di una finestra di dialogo tutta sua, difficilmente personalizzabile. Nel nostro caso useremo un componente DLL già presente in Windows, l' avicap32.dll. AVIcap window class (avicap32.dll) è una libreria di acquisizione che permette di interfacciarsi con dispositivi audio-visivi, senza dover preoccuparci del device,cioè in maniera del tutto indipendente da essi. Infatti la libreria ci offre la completa l'interoperabilità con tutti i dispositivi, l'importante è che siano installaticorrettamente sul sistema, con i dovuti driver caricati. Sul mercato attualmente esistono diversi dispositivi con prestazioni e caratteristiche spesso molto differenti fra loro. Ciascun device ha i suoi driver di installazione e quindi delle routine "proprietarie" di gestione e di utilizzo. Ecco motivata la scelta di utilizzare AVICap. Per intercettare il flusso video in acquisizione dal device è possibile effettuare una chiamata al sistema. Il nostro scopo è quello di ottenere l'accesso ad una porzione della memoria, o meglio ad una finestra di overlay, specificando la posizione, la dimensione e le caratteristiche. Questa finestra di overlay costituirà la nostra area di visualizzazione.
Per eseguire questi passi dovremo lavorare con due DLL di sistema, l'Avicap32.dll e la User32.dll, e per richiamarle è necessario creare delle funzioni che

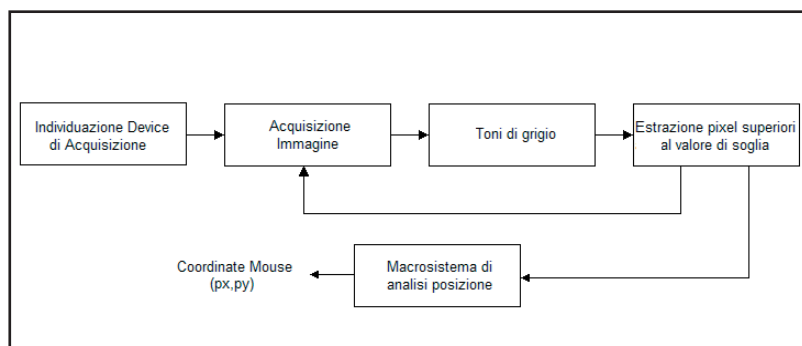



Fig. 2: ***Schema riassuntivo dell'algoritmo di estrazione e di elaborazione del flusso video***

faranno da puntatori alle funzioni interne delle nostre dll. La classe *TrovaVideoDevice* consente di definire una finestra di dialogo da mostrare all'utente e attraverso la quale possiamo elencare tutti i dispositivi individuati nel sistema. A questo punto l-utente dovrà solo scegliere quello che vuole sfruttare per l'applicazione. La classe conterrà una chiamata ad una funzione *capGetDriverDescriptionA* esistente in Avicap32.dll, attraversp la quale è possibile ottenere il riferimento al dispositivo collegato usando semplicemente un indice di posizione. Possiamo definire un metodo che ci ritorna la lista dei device collegati, limitandone la ricerca ai primi 10:

```
using System.Runtime.InteropServices;
...
[DllImport("avicap32.dll")]
protected static extern bool
            capGetDriverDescriptionA(short wDriverIndex,
        [MarshalAs(UnmanagedType.VBByRefStr)]ref
                                    String lpszName,
    int cbName,
      [MarshalAs(UnmanagedType.VBByRefStr)] ref String
                                    lpszVer, int cbVer);

[...]

private static ArrayList GetAllDevices()
    {
      String dName = "".PadRight(100);
      String dVersion = "".PadRight(100);
     ArrayList devices = new ArrayList();
     for (short i = 0; i < 10; i++)
      {
        if (capGetDriverDescriptionA(i, ref
                  dName, 100, ref dVersion, 100))
        {
          devices.Add(dName.Trim());
        }
      }
     return devices;
    }
[...]
```

NOTE

LA FINESTRA DI OVERLAY

La finestra di overlay è una porzione di memoria video destinata alla visualizzazione ed alla cattura di un flusso video proveniente da un dispositivo di acquisizione collegato con il computer. In pratica l'accesso allo stream avviene in maniera similare all'accesso alla memoria, specificandone posizione, dimensione e caratteristiche della superficie di visualizzazione e cattura. La richiesta ha esito positivo soltanto se il sistema vede un driver di acquisizione compatibile fra le risorse disponibili.

Una volta individuato il device di acquisizione da utilizzare, bisogna creare con esso un collegamento per poi procedere alla cattura del flusso video. La dll fornisce gli strumenti per realizzare la connessione, ma poiché per l'elaborazione di ciascun frame del flusso utilizzeremo la libreria OpenCV, per risparmiare qualche passo possiamo sfruttare direttamente le funzioni messe a disposizione dal modulo *highgui*. In particolare, una volta selezionato la posizione del riferimento in memoria del dispositivo, basta effettuare una chiamata al metodo **cvCaptureFromCAM** per allocare ed inizializzare la struttura **CvCapture** che ci consentirà di leggere dal flusso video (al momento consente di interfacciare due videocamere su Windows, attraverso Video for Windows (VFW) e matrox imaging Library, e due su Linux, V4L e FireWire) mediante l'istruzione cvQueryFrame (cattura e restituisce un frame dallo stream sotto forma di IplImage)

```
[...]
CvCapture* cap=cvCaptureFromCAM(trovaDevice-
>Selezionato);
IplImage* frame=cvQueryFrame( cap );
...
```

Sfortunatamente, queste istruzioni sono strettamente legate al sistema: si tratta di codice non gestito e la piattaforma .Net non riesce ad accedere direttamente al puntatore che fa riferimento alla struttura **CvCapture** (vedere il box).
Infatti, già alla prima istruzione otterremo che il puntatore cap è nullo. Per aggirare il problema basta realizzare una managed wrapper class di **CvCapture**

OPENCV
Per informazioni sulla libreria intel OpenCV si può fare riferimento al sito web ufficiale: http://www.intel.com/technology/computing/opencv/index.html

MANAGED WRAPPERS CLASS CON MANAGED C++

Le applicazioni scritte con un linguaggio del framework .net si definiscono "gestite" perche' il framework le fa girare in un ambiente in cui vengono forniti dal Common Language Runtime una serie di servizi come ad esempio gestione della memoria, dei thread, controllo sul codice in esecuzione ed altro. Il concetto di codice gestito è presente in molte altre realtà, come ad esempio Java: la Java Virtual Machine non fa altro che interpretare una serie di comandi generalizzati, uguali per tutte le piattaforme, eseguendole in base al sistema operativo ospite, convertendole in chiamate alle API. Esistono delle soluzioni ibride. L'utilizzo di
In generale, un programma scritto in c# (o magari anche semplicemente in vb6) è sostianzialmente del codice gestito; vengono utilizzate (come nel caso del C#) una serie di librerie, classi e funzioni forniti dall'ambiente di sviluppo che rimappano molte delle funzioni esposte per mezzo delle API di Windows, semplificando mostruosamente la vita ai programmatori. D'altronde, l'aspetto positivo più importante va oltre alla semplice semplificazione: nel caso in cui si dovessero cambiare le API del sistema operativo, l'uso di codice gestito dovrebbe garantire il corretto funzionamento delle applicazioni, in quanto è lo strato d'intepretazione del precompilato che andrebbe sostituito per adattarsi alle modifiche. Lo scopo del framework .net è stato quello consentire lo sviluppo di applicazioni sostanzialmente equiparabili anche usando linguaggi di programmazione differenti, cosi' che non e' piu' indispensabile utilizzare il C++ per creare applicazioni performanti. Naturalmente, le prestazioni delle applicazioni vengono notevolmente influenzate dalla presenza di un ulteriore strato software (il framework). Un codice non gestito è in alcuni casi molto più veloce di un codice gestito. Tuttavia all'interno dei linguaggi .NET esiste una realtà spesso ignorata ma messa a disposizione dei programmatori .NET: Visual C++ .NET e' l'unico linguaggio in grado di integrare codice gestito e non gestito, offrendo soluzioni per sviluppare applicazioni che integrino codice esistente con la nuova piattaforma .NET. Il risultato è una soluzione ibrida.
Nel nostro caso, possiamo incapsulare del codice "nativo" non gestito dal framework attraverso delle wrapper class gestite, appunto, usando Managed C++.

INCAPSULIAMO LA CLASSE CVCAPTURE

La prima cosa da fare è definire una classe di interfacciamento con la struttura **CvCapture** che definiremo unmanaged (non gestita). Attraverso la classe **UMCvCapture** possiamo richiedere l'allocazione del collegamento al device mediante **cvCaptureFromCAM** e quindi l'inizializzazione della struttura **CvCapture**; una volta ottenuto il puntatore alla struttura, su di esso possiamo chiedere ed ottenere il frame corrente del flusso generato. Alla conclusione dell'elaborazione, per rilasciare le risorse allocate bisognerà invocare **cvReleaseCapture**. Bisogna provvedere a dare una definizione del costruttore, del distruttore e di ciascun metodo che vogliamo utilizzare per i nostri scopi.

```
#pragma once
#pragma unmanaged
#include "cv.h"
#include "highgui.h"

class UMCvCapture
{
private:
        CvCapture* capture;
public:
        UMCvCapture(int Selezionato);
        bool Inizializzato(void);
        IplImage* ImmagineCorrente(void);
        void Rilascia(void);
public:
        ~UMCvCapture(void);
};
```

Sarà quindi la classe **UMCvCapture** ad occuparsi dell'interfacciamento diretto con gli strumenti messi a disposizione da OpenCv, in maniera tra l'altro piuttosto semplice come possiamo vedere mostrando l'implementazione dei metodi esposti dalla classe stessa

```
#include "UMCvCapture.h"
UMCvCapture::UMCvCapture(int Selezionato)
{capture = cvCaptureFromCAM(Selezionato );
}
UMCvCapture::~UMCvCapture(void)
{if (capture!=0)
  cvReleaseCapture(&capture);
}
void UMCvCapture::Rilascia(void)
{if (capture!=0)
  cvReleaseCapture(&capture);
}
bool UMCvCapture::Inizializzato(void)
{ return capture!=0;
}
IplImage* UMCvCapture::ImmagineCorrente(void)
{return cvQueryFrame( capture );
}
```

si tratta quindi di singole chiamate a specifiche funzioni già definite.
Definita la classe con codice C++ non gestito, costruiamo adesso la classe di servizio **MCvCapture** con codice gestito da utlizzare per accedere alla struttura dati **UMCvCapture**, facendo corrispondere un metodo gestito a ciascun metodo non gestito, ed in particolare incapsulare il costruttore, il distruttore e ciascuna funzione membro precedentemente definita

```
#pragma once
#include "UMCvCapture.h"
#pragma managed
#using <mscorlib.dll>
using namespace System;
public __gc class MCvCapture
{
public:
        MCvCapture(int selezionato);
        bool Inizializzato(void);
        IplImage* ImmagineCorrente(void);
        void Rilascia(void);
public:
        ~MCvCapture(void);
private:
        UMCvCapture __nogc* m_capture;


};
```

Ciascuna istanza di MCvCapture conterrà al suo interno una definizione di UMCvCapture e quindi una corrispondenza ad una struttura CvCapture.

```
#include "MCvCapture.h"

LightMouse::MCvCapture::MCvCapture(int Selezionato)
{m_capture =new UMCvCapture(Selezionato);
}
LightMouse::MCvCapture::~MCvCapture(void)
{m_capture->~UMCvCapture();
}
void LightMouse::MCvCapture::Rilascia(void)
{m_capture->Rilascia();
}
bool LightMouse::MCvCapture::Inizializzato(void)
{return m_capture->Inizializzato();
}
IplImage* LightMouse::MCvCapture::
                                    ImmagineCorrente(void)
{return m_capture->ImmagineCorrente();
}
```

ACCESSO AL DEVICE CON CODICE GESTITO

Con questa nuova struttura realizzata con codice gestito, l'accesso al device e l'acquisizione dei singoli frame avverrà sostituendo il codice precedente:

```
...
CvCapture* cap=cvCaptureFromCAM(
                        trovaDevice->Selezionato);
IplImage* frame=cvQueryFrame( cap );
...
```

con queste istruzioni equivalenti

```
...
MCvCapture* __capture = new McvCapture
                             (trovaDevice->Selezionato);
IplImage* frame = __capture->ImmagineCorrente();
...
```

Adesso abbiamo tutti gli strumenti necessari all'elaborazione e possiamo dare un corpo all'algoritmo visto in precedenza. Ci connettiamo al device e avviamo un ciclo che ad ogni passo chiede e ottiene un'immagine su cui avviare le analisi (ad esempio il ciclo è implementabile mediante un timer che ad ogni evento di tick programmato preleva il frame e richiama il metodo di elaborazione).

ELABORAZIONE DELL'IMMAGINE

Prelevata il singolo frame dal flusso video, è possibile eseguire una serie di operazioni in presa diretta per realizzare i passi descritti nel diagramma in Fig. 2. Cominciamo convertendo l'immagine originale in toni di grigio, l'immagine *grey* dovrà avere la stessa dimensione dell'originale ma un numero di canali ridotto (passiamo dai 3, RGB, ad uno solo)

DA RICORDARE
Il progetto di esempio allegato all'articolo è stato sviluppato con Visual Studio 2005 e non funzionerà cone le precedenti versioni. Per eseguire l'applicazione di esempio è necessario:
1) Una qualsiasi webcam, ad esempio una Logitec QuickCam. Attenzione però: la distanza a cui ci possiamo mettere da dispositivo è influenzata dalla sua qualità.
2) Un computer non necessariamente ultrapotente, con Windows 2000/XP installato
3) Dot.NET Framework v. 2.0

```
...
IplImage* grey = cvCreateImage(
                         cvGetSize(frame), 8, 1 );
cvCvtColor( frame, grey, CV_BGR2GRAY );
...
```

su *grey* operiamo per isolare i pixel corrispondenti alla sorgente luminosa dal contesto in cui si trova, usando la funzione **DetectLight** così definita

```
...
private: static void DetectLight(IplImage* original,
                         IplImage* modified, int min)
{int nl=modified->height;
 int nc=modified->width;
 uchar* data= (uchar*)(original->imageData);
 uchar* data1= (uchar*)(modified->imageData);
 int num=0;
 for (int j=0; j<nl; j++)
  for (int i=0; i<nc; i++)
  {if (data[num]>=min)
    data1[num]=255;
   else
    data1[num]=0;
   num++;
  }
 return;
}
...
```

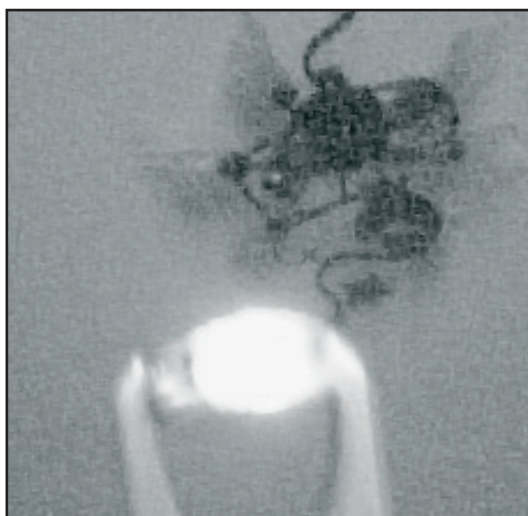



Fig. 3: ***Abbiamo convertito l'immagine da a colori in toni di grigio***

praticamente il metodo effettua un'analisi molto semplice della sequenza di pixel che costituiscono l'immagine originale, valutando il valore di ciascun pixel: se è uguale o superiore al livello di soglia minimo, sull'immagine di risultato inserirà un pixel di colore bianco (valore=255) altrimenti lo sostituirà con uno di colore nero (valore=0). Da notare che l'operazione che stiamo compiendo è di normalizzazione: tutto ciò che è al di sopra della soglia minima diventerà tutto bianco, il resto è totalmente annullato. In questo modo riusciamo a semplificare il passo successivo in quanto si riducono notevolmente tutti i disturbi sull'immagine dovuti ad esempio alla scarsa qualità del dispositivo d'acquisizione. L'immagine risultante [Fig. 4] conterrà soltanto una macchia bianca in corrispondenza dell'area occupata dalla sorgente luminosa sul frame originale: come si può vedere dalla figura, lo sfondo è scomparso e non dovremo più occuparci di esso.

INDIVIDUAZIONE DELLA POSIZIONE

Adesso dobbiamo estrarre il contorno della macchia mediante una chiamata a **cvFindContours**

```
...
  CvSeq* contours = 0;
  CvPoint myCenter;
  CvMemStorage* storage = cvCreateMemStorage(0);
  int header_size = sizeof(CvContour);
  cvFindContours (grey , storage, &contours,
                                          header_size,
          CV_RETR_EXTERNAL,
          CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));
...
```

in contours troveremo tutta la sequenza di punti che individuano l'estremità dei segmenti costituenti il controrno. La sequenza è in realtà spesso suddivisa in blocchi, tenendo conto che il contorno non è perfetto ma abbastanza frammentato, con zone senza informazioni e quindi privo di continuità. Possiamo comunque procedere liberamente

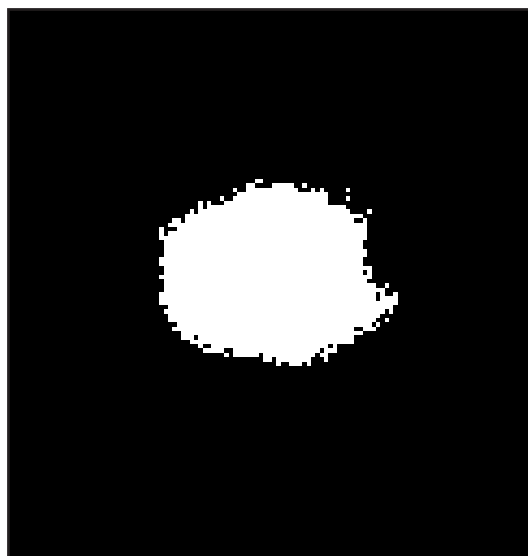



Fig. 4: ***Lo sfondo è stato eliminato e risulta evidenziata soltanto l'area corrispondente alla sorgente luminosa.***

L'AUTORE

Antonino Panella è un ingegnere informatico da anni impegnato nello sviluppo di tecniche basate sulla Computer Vision, anche in ambito mobile su dispositivi Pocket PC. Per informazioni e domande potete far riferimento alla sua mailbox: antonino.panella@gmail.com

NOTE

DLL

Il progetto richiede di inserire il riferimento ai file include e alle librerie di OpenCV per poter compilare correttamente. Una volta generato l'eseguibile, bisogna copiare le dll di servizio rilasciate con il pacchetto di distibuzione della libreria. Troverete tutto il necessario all'interno dello zip allegato alla rivista.

estraendo i punti estremi e cercando un ellisse che possa in qualche modo approssimare il contorno della macchia. Bisogna ricordare che un ellisse è individuabile soltanto se ci sono almento 5 punti noti sul suo perimetro. Per prima cosa verifichiamo che sia possibile definire una curva con i punti del contorno; poi recuperiamo il numero totale di sequenze di punti che definiscono il contorno, allochiamo la memoria per copiare la sequenza dei punti in un array per poi trovare il miglior ellisse che approssima il contorno.

```
...
 if(CV_IS_SEQ_CURVE(contorus))
 { count = circles ->total;
   PointArray = (CvPoint *)malloc(count *
                                   sizeof(CvPoint));
   cvCvtSeqToArray(contorus, PointArray,
                                   CV_WHOLE_SEQ);
   PointArray32f = (CvPoint2D32f *)malloc((count +
                          1) * sizeof(CvPoint2D32f));
   if (count>=6)
   {for (ia=0; ia<count-1; ia++)
    {PointArray32f[ia].x = (float)(PointArray[ia].x);
     PointArray32f[ia].y = (float)(PointArray[ia].y);
    }
    PointArray32f[ia].x = (float)(PointArray[0].x);
    PointArray32f[ia].y = (float)(PointArray[0].y);
    CvBox2D* myBox= (CvBox2D *)
                          malloc(sizeof(CvBox2D));
    cvFitEllipse(PointArray32f, count,myBox);
   .... //continua
}
...
```

con **cvFitEllipse** abbiamo individuato l'ellisse di approssimazione dell'area occupata dalla sorgente luminosa, definito tramite un **CvBox2D**, una struttura che contiene il rettangolo che circoscrive l'ellisse, l'angolo con cui è ruotato rispetto le ascisse e le coordinate del centro rispetto all'immagine. Erano queste le informazioni che andavamo cercando, la posizione del centro (box->center), cioè il punto che idealmente rappresenta la sorgente luminosa all'interno del frame originale.

EVENTI DEL MOUSE

Possiamo adesso legare la posizione del puntatore del mouse sullo schermo del computer con le coordinate del centro individuato all'interno del frame preso dal flusso video. Ogni volta che terminiamo l'elaborazione con un risultato positivo (cioè si è accertata la presenza della sorgente luminosa e si è individuato il centro dell'area occupata da essa sull'immagine), simuliamo l'evento globale di movimento del mouse usando la funzione mouse_event delle API di Windows.

```
...
VOID mouse_event(DWORD dwFlags,
    DWORD dx,
    DWORD dy,
    DWORD dwData,
    ULONG_PTR dwExtraInfo
);
...
```

il primo parametro corrisponde alla categoria dell'evento del mouse (movem leftclick_up, rightclick_down, etc), il secondo ed il terzo contengono lo scostamento assoluto rispetto alla precedente posizione del puntatore, dwData è legato alla rotazione della rotella mentre dwExtraInfo conterrà informazioni aggiuntive. Ad esempio, un evento può essere inviato così

```
...
 mouse_event(MOUSEEVENTF_MOVE, dx, dy, 0, 0);
...
```

Ad ogni ciclo di elaborazione ci teniamo sempre in memoria la posizione calcolata al passo precedente, in modo da calcolare sempre la differenza da inviare come evento di mouse. Il risultato è molto importante, in quanto inviamo al sistema operativo dei messaggi che verranno interpretati come prodotti dal mouse stesso e quindi gestiti di conseguenza, senza doverci occupare di altro. Quello ottenuto è uno strumento in grado di convertire il movimento di una luce in messaggi uguali a quelli del mouse.

Antonino Panella

COSA VUOL DIRE IPLIMAGE?

È l'acronimo di Image Processing Library – Image ed è il formato standard scelto da Intel per rappresentare un'immagine nelle API IPL e OpenCV. Tramite questa struttura dati è possibile gestire completamente tutto cioè che ruota intorno alle immagini: caricamento e salvataggio, conversione di formato, elaborazione, filtraggio ed infine visualizzazione.

SUGGERIMENTI

L'uso di LightMouse può essere per alcuni un po' complicato, specialmente a chi non ha mai usato il touchpad dei portatili. Non demordere, bisogna prenderci un po' la mano prima di riuscire ad ottenere un movimento fluido. Ricordate che spesso basta agire sui parametri della finestra di controllo per risolvere tante difficoltà, ma ricordate sempre che in caso di necessità basta semplicemente spegnere la luce o tappare la lente della webcam per ritornare al vecchio e caro mouse. LightMouse non ha la pretesa di essere un vero e proprio servizio di sistema: è un semplice programma eseguito in background, pronto a fornirci a richiesta i suoi servizi, in modo da non creare conflitti con il mouse. Se lo si volesse avere sempre pronto, basta creare un collegamento nella cartella Startup di Windows a LightMouse.exe. Per avviarlo, andate sulla System Tray e scegliete Avvia.
Mettete a fuoco la webcam, evitate di riprendere lampadari, finestre o superfici che potrebbero produrre riflessi e riverberi, magari abbassate un po' le serrande per migliorare la resa (oltre che creare un po' di atmosfera).

BORLAND RITORNO CON IL TURBO!!!

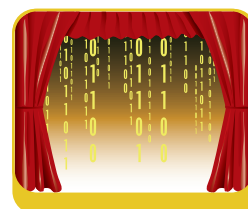
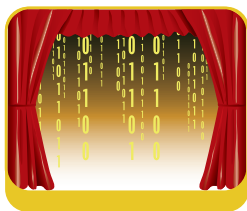
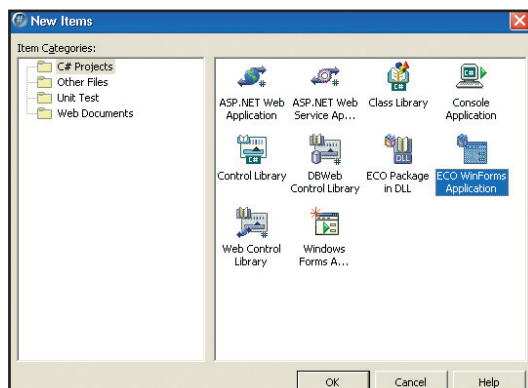
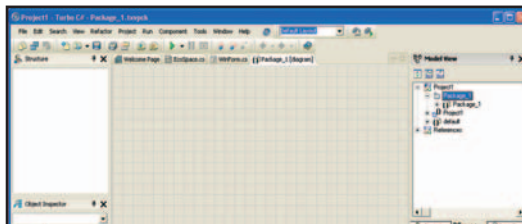
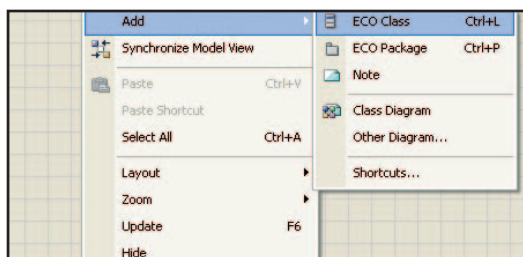
DOPO ESSERSI DEDICATA PER ANNI AL MERCATO DEGLI ALM, BORLAND CREA UNA NUOVA DIVISIONE PER IL RILANCIO DEI COMPILATORI E DEGLI IDE DEDICATI ALLA PRODUTTIVITÀ INDIVIDUALE. NASCONO I PRODOTTI TURBO EXPLORER E SONO DAVVERO ENTUSIASMANTI

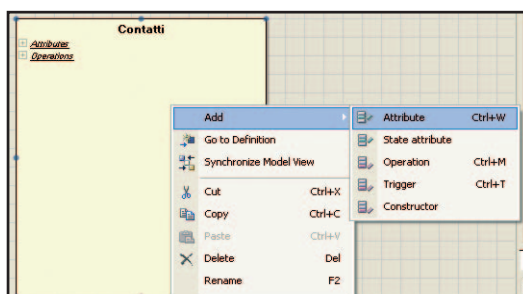
Borland è tornata. Non che fosse mai sparita, ma la società si era concentrata principalmente sullo sviluppo di soluzioni ALM, Application Lifecycle Management. Questa scelta strategica aveva portato a sacrificare gli investimenti verso un settore storico di Borland, quello degli IDE e dei tool di sviluppo. Eppure il "vecchio" Turbo Pascal ha segnato l'inizio del boom della programmazione ad oggetti, così come Delphi è stato il capostipite degli IDE RAd. Proprio per non perdere questo patrimonio e per tornare ad occuparsi, in modo primario di questo settore storico, l'8 Febbraio scorso Borland aveva annunciato l'intenzione di trovare un acquirente per la linea di prodotti IDE. L'acquisto coinvolgerebbe Delphi, C++Builder, C#Builder, JBuilder (e Peloton), InterBase, JDataStore, nDataStore e la linea dei prodotti Turbo. L'obiettivo della nuova società è quello di concentrare il proprio Focus sulla produttività dello sviluppatore individuale. Il primo passo verso la formazione della nuova società è stato quello di creare una divisione apposita: la Devco. Alla quale partecipano tutti gli attori storici dello sviluppo degli IDE Borland capeggiati dal mitico David Intersimone, personaggio carismatico del gruppo a cui si deve l'esistenza stessa di buona parte dei prodotti Borland. La nuova divisione dovrebbe essere ceduta ad un'acquirente, sono al vaglio molte ipotesi, quello che più conta è che il gruppo non sarà ceduto ad una società informatica ma semplicemente ad un soggetto "finanziario", in grado di supportare lo sviluppo tecnologico dei vari prodotti. Il messaggio è forte e chiaro. La nuova divisione vuole solo fare ottimi prodotti rivolti agli sviluppatori e non vuole avere interferenze dovute a scelte strategiche e/o finanziarie. Insomma quelli di David Intersimone vogliono occuparsi di ciò che gli piace fare: sviluppare.

I PRODOTTI

Il primo passo del nuovo Developer Tools Group di Borland è stato quello di rilasciare una nuova versione di ogni singolo prodotto compreso nel Borland Developer Studio e raggruppare questa nuova serie di prodotti sotto l'etichetta "Turbo", come a voler dire: "ricominciamo dove abbiamo lasciato tanti anni fa, dai prodotti migliori che abbiamo mai fatto". La nuova serie di Ide comprende: Turbo Delphi® per Win32, Turbo Delphi per .NET®, Turbo C++® e Turbo C#®. Ogni versione è disponibile in 2 versioni: Turbo Explorer, gratuita disponibile per il download e Turbo Professional, una versione con un prezzo inferiore a 500 , pensata per poter lavorare con migliaia di tool di terze parti, componenti e plug-in. Entrambe le versioni Turbo consentono agli sviluppatori di creare rapidamente GUI, database e Web Services ad alte performance per Microsoft Windows. Turbo Delphi per .NET e Turbo C# supportano Microsoft .NET e ASP.NET.

L'APPLICAZIONE DI ESEMPIO

In questo primo articolo abbiamo deciso di occuparci di Turbo C#. Lo faremo, come sempre, con un approccio molto pratico ovvero mostrando quella che è una delle caratteristiche vincenti dei prodotti Borland di ultima generazione: ECO.
ECO è l'acronimo di Enterprise Core Object. Non spaventi questa definizione. Vedremo come ECO sia infatti principalmente uno strumento per accelerare ulteriormente lo sviluppo delle applicazioni. Questa volta la logica di programmazione sarà leggermente diversa rispetto a quella classica degli IDE , che tipicamente partono dallo sviluppo dell'interfaccia onde poi programmarne il comportamento in reazione a determinati eventi. Questa volta inizieremo modellando una serie di classi per poi "associarle" all'interfaccia. Il nostro scopo sarà realizzare una piccola agendina dei contatti. La nostra agendina prevede lo sviluppo di una classe base: Contatto, e di due classi astratte derivate dalla prima, rispettivamente Persona e Società. La classe Contatto sarà definita da 3 campi, rispettivamente

VIDEOGUIDA TURBO AVI

NomePersona, NumeroTelefonico, Città. Proviamo a realizzare il tutto con ECO. Il primo passo sarà creare una nuova applicazione di tipo “EcoWinForms Application”.

Dopo pochi secondi otterrete lo scheletro di un’applicazione funzionante. Le caratteristiche sono quelle tipiche di un ambiente RAD. C’è la finestra delle proprietà, c’è la gestione degli eventi, ci sono una serie di file che contengono il codice che viene mascherato dall’IDE. Rispetto a quanto conosciamo normalmente c’è anche una tabsheet che indica “Model View”, è su questa che dobbiamo cliccare per iniziare il nostro lavoro di modellazione delle classi.

Successivamente cliccheremo due volte su “Package1” si avvierà il modellatore delle classi. Utilizziamo il pulsante destro del Mouse e dal menu a tendina scegliamo “Add Eco Class”.

La finestra appena comparsa rappresenta la nostra classe. Rinominiamola “Contatti”.
Agendo ancora una volta sul pulsante destro del mouse scegliamo “Add Attribute”, e aggiungiamo rispettivamente tre attributi:

NomeCompleto: String
Città: String
Numerotelefonico: String

Nella finestra delle proprietà dichiariamo la classe come “Astratta”. A questo punto aggiungiamo esattamente come nel caso precedente le due classi “Persona” e “Società”. Non sarà necessario né dichiararle come astratte né settare attributi. Semplicemente le collegheremo alla classe “Contatti” per mezzo di un componente “*Generalization Implementation*”.
Proviamo a compilare, e se tutto e andato a buon

INSTALLARE TURBO C#

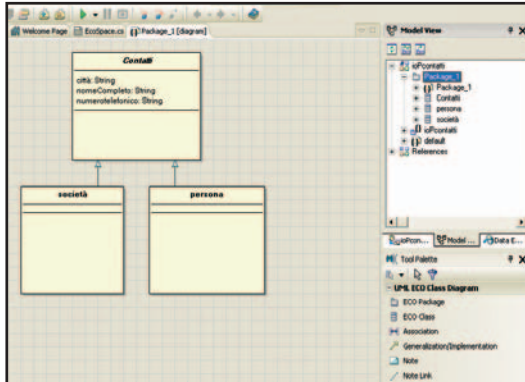
Prima di tutto dovete dotarvi di due file. Uno contentente il setup di Turbo C#, l’altro contenete i prerequisiti necessari all’installazione. Inoltre avrete bisogno di una chiave di attivazione per il prodotto. All’interno del cd allegato ad ioProgrammo di questo mese troverete il file di installazione di Turbo C# Explorer, mancano invece il file dei prerequisiti per questioni legate alla licenza del software prodotto da terze parti in esso contenute e la chiave di attivazione del prodotto, per ovvi motivi. Prima di ogni cosa è necessario puntare il proprio browser all’URL: http://www.borland.com/downloads/download_turbo.html. A questo punto avete a disposizione due strade:

1) Non disponete del file dei prerequisiti oppure del file di installazione del prodotto
2) Disponete di tutto quello che vi serve ma avete bisogno di una chiave di attivazione.

Nel primo caso selezioniamo “Turbo C# explorer”, effettuiamo la registrazione e nella maschera successiva selezioniamo i file che ci servo per il download. Scegliamo “Full prerequites” nel primo gruppo di file e la lingua inglese per l’installazione del prodotto. In entrambi i casi ci sarà inviata una mail. Controlliamo di averla ricevuta correttamente e prendiamone nota mentalmente, il suo contenuto ci sarà utile in un secondo momento. Una volta che i file di installazione e prerequisiti saranno in vostro possesso dovremo “scompattare” il loro contenuto in una directory temporanea e installare nell’ordine:

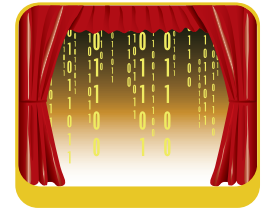
- Il dot net framework redist
- il dot net framework Sdk
- il dot net C#
- msxml 4.0
- Internet explorer 6.0 nel caso improbabile che non l’abbiate

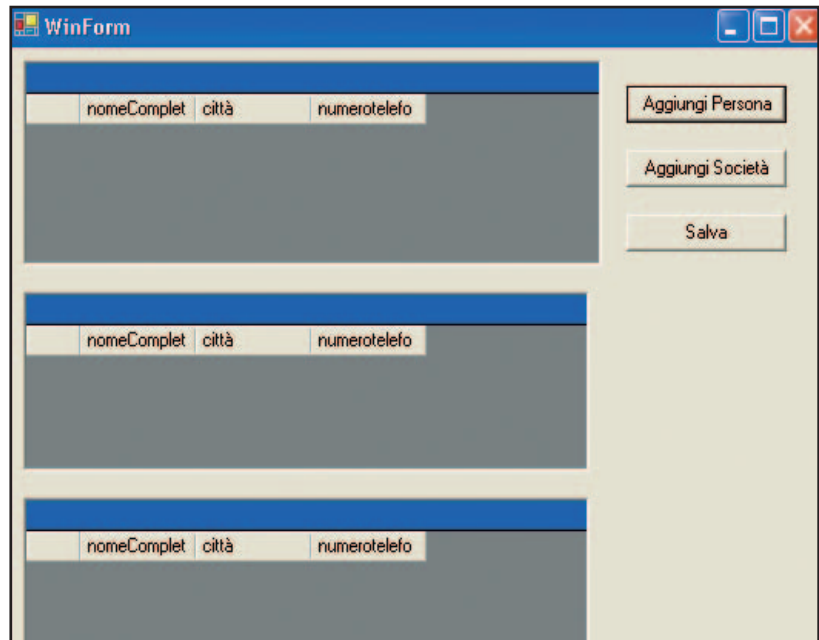
Una volta fatto questo potete procedere all’instalazione di Turbo C# Explorer. Per questo è sufficiente seguire l’installazione guidata. Al termine dell’installazione dovremo attivare il prodotto inserendo il file di licenza. Ripescate la mail che Borland vi aveva mandato e contenente il codice di attivazione. Al suo interno troverete le indicazioni relative al percorso in cui copiare il file di attivazione della licenza. Copiate il file in questione in una directory compatibile con il vostro sistema e infine avviate Turbo Explorer C#. La vostra versione gratuita di Turbo C# Explorer è pronta.

fine otterremo una form vuota. L'unica cosa che possiamo annotare è che il compilatore è veramente velocissimo. Se controlliamo nel code explorer, noteremo però che ECO ha generato per noi tutta la serie completa del codice che implementa le classi che abbiamo appena disegnato.

DISEGNIAMO L'INTERFACCIA

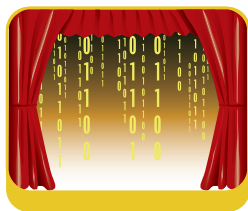
Dobbiamo ora "Agganciare" i componenti dell'interfaccia alle nostre classi. Per farlo abbiamo bisogno di un po' di componenti. In particolare:
due bottoni

AggiungiPersona	
	Tipo: button
	Name: aggiungiPersona
	Text: aggiungi persona
AggiungiSocietà	
	Tipo: button
	Name: aggiungiSocietà
	Text: aggiungi società
ehContatti	
	Tipo: expressionHandle
	Root: rhroot
	Expression : contact.allinstances
ehPersona:	
	Tipo: expressionHandle
	Root: ehRoot
	Expression: person.allinstances
ehSocietà:	
	Tipo: expressionHandle
	Root: rhroot
	Expression: società.allinstances
grigliaContatti	
	Tipo: datagrid
	Datasource: ehContatti

grigliaPersona	
	tipo: datagrid
	Datasource: ehPersona
grigliaSocietà	
	Tipo: datagrid
	Datasource: ehSocietà

Sorvoleremo in questa sede sul senso dei vari *ExpressionHandle*, li considereremo in questo momento come dei wrapper che incorporano tutti i metodi offerti dalle classi create in precedenza.
Se tutto è impostato in maniera corretta, provando a compilare otterremo qualcosa di simile a quanto mostrato in figura
In sostanza le tre griglie hanno "Agganciato" i campi che abbiamo creato precedentemente nel "model". Tuttavia al momento non è possibile ancora inserire dati in griglia. Per supportare questa operazione possiamo settare le proprietà dei bottoni *"AggiungiSocietà"*, *"AggiungiPersona"*, come segue:

BindingContext: grigliasocietà
Ecolist: Add
RootHandle: ehSocietà

Fatto questo possiamo provare a ricompilare. Noterete che è adesso possibile aggiungere campi alle griglie. I valori riportati per le griglie società e persona compariranno anche nella griglia Contacts.

LA PERSISTENZA

Per implementare la persistenza avremo bisogno di compiere pochi passi:

- cliccare due volte sul nome del file Ecospace.cs
- aggiungere un componente Persistence MapperXml
- settare la sua proprietà filename a dati.xml
- ciccare sullo spazio bianco sottostante e nella scheda delle proprietà settare persistence Mapper con il nome del nuovo componente.

Infine aggiungiamo un bottone “update” e cliccando due volte su di esso definiamo l’evento OnClick con il codice seguente:

```
Ecospace.UpdateDatabase
```

Proviamo a ricompilare...ed il gioco è fatto, questa volta i dati saranno salvati e la persistenza implementata. La cosa interessante è che non ci siamo neanche preoccupati di controllare il formato con cui questi dati sono stati salvati in XML. Ma siamo convinti che si tratti di uno standard.

CONCLUSIONI

Per realizzare il nostro progetto abbiamo impiegato non più di 13 minut e una sola riga di codice. Si tratta di una rubrichetta dei contatti con persistenza dei dati in un file XML. La cosa interessante consiste proprio nella possibilità di “Modellizzare” il progetto prima di realizzarne l’interfaccia. La nuova linea di Borland promette più che bene. C’è ancora qualche elemento da mettere a punto, ad esempio attualmente tutto funziona con la versione 1.1 del .NET framework, ma probabilmente non è intenzione di Borland agganciarsi troppo agli strumenti già implementati da Microsoft. Ma questo rientra in considerazioni commerciali e noi come il Dev Group di Borland non abbiamo nessuna intenzione di lanciarci in questo genere di considerazioni. Quello che ci preme è valutare l’aspetto tecnico di un prodotto, e dal punto di vista squisitamente tecnico il primo rilascio della nuova serie dei prodotti Borland è decisamente ottimo.

LETTERA DI DAVID INTERSIMONE AGLI SVILUPPATORI

Ho iniziato usando Turbo Pascal 1.0 nel novembre del 1983, quando Philippe Kahn mi diede una copia al Comdex di Las Vegas. Dopo averlo installato sul mio PC IBM ho capito immediatamente che era qualcosa di diverso. Quel giorno ho capito di voler andare a lavorare per Borland. Ho iniziato a lavorare in Borland il 17 giugno 1985 e negli ultimi 20 anni ho avuto il piacere di far parte di una fantastica società e grande comunità di sviluppatori software. Ho avuto il lusso e il piacere di lavorare nell’R&D nei primi anni di vita dei linguaggi Turbo. Negli ultimi 15 anni ho tenuto le relazioni con gli sviluppatori, viaggiando in tutto il mondo per incontrare decine di migliaia di programmatori. Ogni giorno vado al lavoro e collaboro con i migliori ingegneri del software al mondo che più si focalizzano sulle necessità dello sviluppatore.
Sono molto eccitato di passare alla nuova società. Abbiamo le giuste persone, abbiamo il giusto eco-sistema di prodotti e componenti di terze parti, abbiamo gli autori, gli insegnanti, i consulenti, gli esperti e una comunità fedele – il componente più importante. La nostra nuova società sarà completamente focalizzata su di voi e il vostro successo. Si, entrambe le società si focalizzeranno sullo sviluppo software. Entrambe lavoreranno per avanzare lo stato dell’arte del software. Lo faranno semplicemente in due modi differenti. La nostra lo farà dedicandosi sulla produttività del singolo sviluppatore e creando prodotti di qualità per lo sviluppo applicativo, utilizzando i nostri pluripremiati IDE, strumenti, librerie di componenti, classi e database. Borland lo farà indirizzando le necessità di più grandi organizzazioni, aiutandole a ottimizzare la delivery dei loro progetti software.

Daryl Taft di eWeek mi ha chiesto: “Come impiegato più “anziano” di Borland, come ti colpisce lo spin-off?” Ho risposto dicendo che vado verso la nuova società con un gran sorriso sulla faccia e una piccola lacrima negli occhi. Voglio assicurare a tutti voi che siamo qui a Scotts Valley, e in tutto il mondo, Italia compresa, per lavorare sulle future versioni di Delphi, JBuilder e di tutti gli altri prodotti. Stiamo ascoltando, ancora e di più, le vostre necessità, problemi e suggerimenti. Stiamo seguendo e valutando le tecnologie relative a Windows, .NET, Java, open standard e le tecnologie emergenti da cui volete e vorrete trarre vantaggio.

Sono convinto che questa sia la cosa giusta da fare per il nostro comune business: gli IDE. Ed è la cosa giusta per l’ALM di Borland. La nostra priorità ora è quella di una migrazione riuscita e non traumatica per i nostri sviluppatori e il creare una macchina con maggiori investimenti, unica focalizzazione e forte crescita. Questa non è la fine di una linea di prodotto ma il suo il potenziamento. Questa mossa costituisce la migliore garanzia per i nostri clienti, la nostra comunità e la nostra società stessa.

Il gruppo dei tool di sviluppo in Borland ha già costituito una divisione completamente indipendente, con personale e budget dedicati, che opera in maniera autonoma per il bene della comunità. Non abbiamo ancora identificato l’acquirente della divisione. Posso anticipare che non sarà alcuna società di software o hardware e che sto personalmente lavorando con il top management per identificare il giusto acquirente, che porterà avanti gli interessi degli IDE. Ricordate: Borland è impegnata prima di tutto verso i suoi clienti e le loro necessità. L’8 febbraio mi hanno ridato la “mia” società.

Go Borland. Go New Company. Go Developers!

David Intersimone, "David I" Vice President, Developer Relations e Chief Evangelist Borland Software Corporation

LASZLO QUASI COME FLEX MA GRATIS

C'È UN FILE DI TESTO CONTENENTE XML, C'È UN COMPILATORE ON THE FLY. TU RICHIAMI UNA PAGINA WEB, IL COMPILATORE INTERPRETA IL FILE XML E TIRA FUORI UN'APPLICAZIONE FLASH! TUTTO GRATUITO, TUTTO SUPERCOMPATIBILE...WOW!

Con la sigla Web 2.0 non si fa altro che identificare la seconda ondata evolutiva che sta interessando le applicazioni web. Questa tipologia di applicazioni sono convenzionalmente caratterizzate da un'interazione, che si è soliti definire, di tipo *click, wait and refresh*. Il normale flusso delle operazioni è, infatti, definito da un'interazione utente (click) che è comunicata ad un web server. Quest'ultimo elabora l'operazione e risponde con l'invio di una nuova pagina (wait) che è visualizzata sul browser (refresh). Da un punto di vista di basso livello si parla di modello di comunicazione sincrona con flusso logico di tipo page-driver, determinato lato server. Solitamente l'utente percepisce una scarsa usabilità, per niente paragonabile a quella che si ha con l'utilizzo delle applicazioni desktop, che hanno tempi di risposta minori e feedback utente legati alle attività più lunghe. Tali lacune, insieme con altri fattori come la riduzione dei tempi di download e l'evoluzione dei web browser, hanno fatto nascere una nuova tecnologia denominata Rich Internet Applications (RIA), le cui principali caratteristiche sono: interazione immediata senza percezione di roundtrip del server, logica applicativa suddivisa fra client e server ed esecuzione locale in un ambiente sicuro denominato sandbox. Il beneficio più significativo, legato all'uso di tale tecnologia, è dato da un maggiore impatto dello strato di presentazione il cui scopo primario è accrescere quello che in termini economi è chiamato ROI: *return on investment*. AJAX è attualmente la tecnologia RIA più diffusa (vedi Gmail e GoogleMaps). Nel corso di questo articolo introdurremo OpenLaszlo, un framework J2EE per lo sviluppo di applicazioni RIA che rappresenta l'alternativa open-source al più blasonato e commerciale Flex, sviluppato da Macromedia, di cui ci siamo già interessati nel numero106 di ioProgrammo.

Fig. 1: ***Una RIA dimostrativa rappresentante il catalogo musicale del sito Amazon***

FLUSSO APPLICATIVO E CONCETTI BASE

La distribuzione di OpenLaszlo, utilizzata in questo numero, include la versione 5.0.24 di Tomcat. È possibile integrare la piattaforma OpenLaszlo all'interno di un qualsiasi J2EE servlet container. Il flusso applicativo è abbastanza semplice in quanto analogo a quello presentato per le JSP (Java Server Pages):

1. L'utente, attraverso il browser, contatta un URL che punta ad un'applicazione OpenLaszlo inviando una richiesta http
2. La piattaforma OpenLaszlo individua il file in formato XML e lo compila solo nel caso in cui il file non sia mai stato compilato o abbia subito modifiche dalla sua ultima compilazione.
3. L'applicazione viene ritornata al web browser sotto forma di byitecode SWF (versione 8).

Dal punto di vista di uno sviluppatore, la parte più interessante è rappresentata dal codice sorgente. Le applicazioni OpenLaszlo sono definite da uno o più file in formato XML (well-formed), che convenzionalmente riportano l'estensione .lzx, contenenti un linguaggio di programmazione *object-oriented* basato su tag e facente uso di sintassi XML e JavaScript. Tale linguaggio, detto LZX Programming Language, essendo molto

REQUISITI

Conoscenze richieste
basi di JavaScript, conoscenza di XML

Software
Java 2 Standard Edition SDK 1.4.0 o superiore, Eclipse 3.1 SDK + Eclipse Web Standard Tools (WST) 1.0.0 o superiore

Impegno

Tempo di realizzazione

simile a XUL (XML User interface Language), induce lo sviluppatore all'uso del paradigma MVC (Model-View-Controller). Come primo esempio non potevamo esimerci dal creare il classico helloworld.lzx:

```
<?xml version="1.0" encoding="UTF-8" ?>
<canvas>
        <text x="177" y="172" id="label1"
            width="113" fontsize="19"
            font="Arial"
            fgcolor="0x000000"
            fontstyle="bold"
            text="Hello World!!!">
        </text>
</canvas>
```

Per compilare ed eseguire è sufficiente copiare il file nella cartella *<OPENLASZLO_SERVER_ INSTALL_ DIR>\Server\lps-3.3.1\my-apps* e puntare, via web browser, l'URL http://localhost:8080/lps-3.3.1/my-apps/helloworld.lzx: l'applicazione visualizzeràil più classico degli "Hello World!!!". Come si può notare si tratta di un comune documento XML il cui elemento più interessante è il tag *text*, le cui proprietà ne definiscono la visualizzazione a video.

GESTIONE DEGLI EVENTI

In questa sezione aggiungeremo al precedente esempio un pulsante dotato di logica applicativa:

```
<?xml version="1.0" encoding="UTF-8" ?>
<canvas>
        <text x="177" y="172" id="label1"
            width="113" fontsize="19"
            font="Arial"
            fgcolor="0x000000"
            fontstyle="bold"
            text="Hello World!!!" resize="true">
        </text>
        <button text="Button" id="button1"
             onclick="label1.setText('Siamo tutti
                         amici di ioProgrammo')"
             x="210" y="194">
        </button>
</canvas>
```

La proprietà *onclick*, del componente identificato dal tag *button*, specifica l'azione da eseguire a fronte dell'evento di click: il cambio della *label1* da "Hello World!!!" a "Siamo tutti amici di ioProgrammo" invocando il metodo *setText*.
Il codice appena visto è perfettamente funzionante ma risulta poco leggibile. Per migliorarne la forma e renderlo più elegante potremmo disaccoppiare la logica dell'azione da quella del componente:

```
<button text="Button" id="button1"
                onclick="changeLabel()"
                x="210" y="194">
        </button>
        <script>
            function changeLabel() {
                label1.setText('Siamo tutti amici di
                                    ioProgrammo')
            }
        </script>
```

APPLICAZIONI ANIMATE

Le animazioni all'interno delle applicazioni sono uno strumento estremamente utile per quanto riguarda la costruzione di interfacce utente accattivanti. In questo paragrafo vedremo come sia possibile inserire semplici animazioni seguendo i metodi messi a disposizione delle API di OpenLaszlo. Quello che faremo è creare un'applicazione contenente una label. Cliccando su questo componente sarà aperto un nuovo pannello che si espanderà gradualmente ed al cui interno saranno presenti un messaggio ed un bottone:

```
<canvas>
    <view width="200" bgcolor="#345467">
        <simplelayout axis="y"/>
        <text fgcolor="red" fontstyle="bold"
                    clickable="true">ioProgrammo
            <method event="onclick">
parent.details.animate("height",parent.details.height
                        ==0?200:0, 1000, false);
            </method>
        </text>
        <view name="details" height="0"
                width="${parent.width}" clip="true" >
            <text y="10" multiline="true"
        fgcolor="white" width="${parent.width}">LA PIU'
 DIFFUSA RIVISTA DI PROGRAMMAZIONE!<br></br>
  Se sei un appassionato di programmazione questa è
                    la rivista che fa per te!</text>
            <button y="90" text="Clicca per entrare!">
                <method event="onclick">
getURL("javascript:NewWindow=window.open('http://
                  www.ioprogrammo.it','newWin')")
                </method>
            </button>
        </view>
    </view>
</canvas>
```

La parte più significativa di questo esempio risiede

nell'*onclick* del componente *text*. L'azione associata invoca il metodo animate, presente in tutti i componenti messi a disposizione dal framework, variando l'altezza del componente *details* da 0 a 200 pixel in un tempo di 1000 ms.
Se avete fatto attenzione, avrete anche notato la terza e ultima modalità per la definizione dei comportamenti legati ad un dato evento: attraverso il tag *method*. Analogamente alle classi Java, i componenti forniti da OpenLaszlo sono composti da attributi e metodi. Quest'ultimi possono essere ridefiniti come nell'esempio dell'*onclick*. Per maggiori informazioni relativi agli oggetti LZX è possibile consultare la reference LZX contenuta nel pacchetto d'installazione.

DATA BINDING

Negli esempi finora descritti, i dati presentati all'utente sono stati esplicitamente dichiarati nei sorgenti. Questa metodologia, oltre ad essere poco elegante, necessita la modifica e la ricompilazione del codice ogni volta i dati vengano modificati. Per separare la parte dati dal codice sorgente, OpenLaszlo mette a disposizione dello sviluppatore delle utili funzioni di databinding attraverso le quali è possibile associare ai componenti LZX il contenuto di strutture dati definite, tanto per cambiare, in formato XML. Ad esempio, il seguente documento XML rappresentante il contenuto di un'ipotetica videoteca:

```
<?xml version="1.0" encoding="UTF-8"?>
<videoteca>
        <film>
                <titolo>Spider-man</titolo>
                <regista>Sam Raimi</regista>
                <anno>2002</anno>
        </film>
        <film>
                <titolo>La vita è bella</titolo>
                <regista>Roberto
                            Benigni</regista>
                <anno>1998</anno>
        </film>
        <film>
                <titolo>Titanic</titolo>
                <regista>James
                           Cameron</regista>
                <anno>1998</anno>
        </film>
</videoteca>
```

può essere "collegato" ad un'applicazione OpenLaszlo usando il tag dataset:

```
<canvas>
   <dataset name="dataset" src="videoteca.xml"/>
   <simplelayout axis="y"/>
   <view datapath="dataset:/videoteca/film">
      <simplelayout axis="x"/>
      <text datapath="titolo/text()"/>
      <text datapath="regista/text()"/>
      <text datapath="anno/text()"/>
   </view>
</canvas>
```

Il contenuto del documento XML è "mappato" nell'elemento *dataset* ed è successivamente utilizzato nei componenti *text* attraverso la proprietà *datapath*. Sia per la parte di binding sia per quella di accesso ai dati, OpenLaszlo utilizza un sottoinsieme di funzioni XPath, lo standard definito dal consorzio W3C per identificare gli elementi all'interno di documenti XML.
L'esempio appena visualizzato descrive come integrare una struttura dati statica. Adesso vedremo come lo stesso documento XML possa essere generato dinamicamente attraverso una chiamata server-side aggiungendo degli specifici attributi al tag *dataset*:

```
<dataset name="dataset"
      autorequest="true" type="http"
      src="getVideoteca.jsp"/>
```

In questo modo stiamo chiedendo al *dataset* di "mappare" la struttura dati attraverso la chiamata di una pagina JSP residente sul server:

```
<%@ page import="java.sql.*"%>
<videoteca>
<%
   Connection connection = null;
   try {
      Class.forName("org.hsqldb.jdbcDriver");
      connection = DriverManager.getConnection
         ("jdbc:hsqldb:file:videotecastore", "sa", "");
      Statement stmt =
                     connection.createStatement();
      ResultSet rs = stmt.executeQuery("select *
                              from videoteca");
      while (rs.next()) {
   %>
         <film>
            <titolo><%=
               rs.getString("titolo")%></titolo>
               <regista><%=
              rs.getString("regista")%></regista>
               <anno><%=
                  rs.getInt("anno")%></anno>
           </film>
   <%
      }
```

```
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {}
    }
%>
</videoteca>
```

In questo esempio si è scelto di utilizzare una pagina JSP che raccoglie i dati da un database e li ritorna formattati in XML, ma è possibile utilizzare qualunque tipo di tecnologia server-side capace di generare dinamicamente documenti XML.

Fig. 2: *Uno screenshot che mostra il Laszlo Debugger*

LASZLO DEBUGGER

In questo paragrafo vedremo come OpenLaszlo viene incontro ad una delle fondamentali necessità dello sviluppatore: il debugging. Il framework, infatti, mette a disposizione il Laszlo Debugger un utile tool per ispezionare gli stati dell'applicazione attraverso delle stampe. Nel seguente codice viene riproposto il sorgente usato per la descrizione della gestione eventi in modalità debug:

```
<?xml version="1.0" encoding="UTF-8" ?>
<canvas debug="true">
        <text x="177" y="172"
            width="113" fontsize="19"
            font="Arial"
            fgcolor="0x000000"
            fontstyle="bold"
            text="Hello World!!!" resize="true"
                                id="label1">
        </text>
        <button text="Button" id="button1"
            onclick="changeLabel()"
            x="210" y="194">
        </button>
        <script>
          function changeLabel() {
            Debug.write('esecuzione della
                    funzione changeLabel()');
            label1.setText('Siamo tutti amici di
                        ioProgrammo')
          }
        </script>
</canvas>
```

Per abilitare il debugger, come mostrato in **Figura 2**, è stato necessario impostare a true la proprietà debug del tag canvas. Mentre, per stampare nella debugger console è sufficiente aggiungere l'invocazione del metodo Debug.write.

LASZLO DENTRO ECLIPSE

La consistenza della soluzione OpenLaszlo è misurabile anche dall'attenzione che un colosso internazionale come IBM sta dedicando a questa tecnologia. Da Ottobre 2005, IBM e Laszlo Systems hanno infatti raggiunto un accordo per lo sviluppo di un plugin per Eclipse il cui compito è quello di facilitare la creazione ed il testing di progetti basati su OpenLaszlo. Il plugin, distribuito gratuitamente, è dotato di interessanti caratteristiche quali XML Syntax Highlighting e Script-based content assistance. Inoltre, attraverso la vista "Preview" è possibile eseguire l'applicazione su cui stiamo lavorando senza il bisogno di fare deploy sul web server.

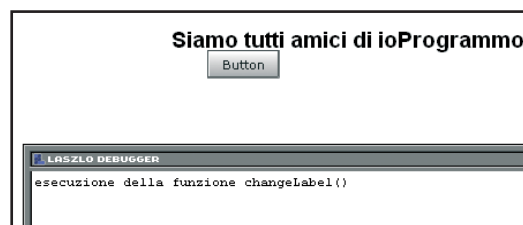



Fig. 3: *Uno screenshot che mostra la preview di un'applicazione OpenLaszlo*

CONCLUSIONI

Le Rich Internet Applications rappresentano un nuovo modo di programmare e sono particolarmente consigliate per applicazioni contenenti processi multistep o che necessitano di validazione lato client. La loro introduzione permette un notevole passo in avanti in termini di interazione utente, ma allo stesso tempo aggiunge delle complessità aggiuntive. In questo articolo abbiamo visto come attraverso l'uso di OpenLaszlo, un presentation server open-source, sia possibile implementarle in maniera facile e veloce.

Fabrizio Fortino

JSON: IL WEB DI NUOVA GENERAZIONE

IN QUESTO ARTICOLO APPRENDEREMO ALCUNE TECNICHE CHE CI CONSENTIRANNO DI REALIZZARE FACILMENTE APPLICAZIONI AJAX-BASED. VEDREMO, INOLTRE, CHE XML NON SEMPRE COSTITUISCE LA SOLUZIONE MIGLIORE PER L'INTERSCAMBIO DI DATI

JSON è un formato utilizzato per rappresentare e scambiare dati. In pratica, svolge la stessa funzione di XML con il vantaggio di essere più leggero. Inoltre, utilizzando JavaScript, è molto semplice effettuare il parsing di un messaggio in formato JSON. Addirittura vedremo che è sufficiente una sola istruzione! JSON è l'acronimo di JavaScript Object Notation. Una stringa in formato JSON non è altro che un oggetto e/o un array JavaScript espresso in modo letterale.
AJAX è la tecnologia emergente che ha rivoluzionato il modo di concepire la programmazione Web. La rivoluzione è stata tale da riferirsi spesso ad essa col termine Web 2.0. AJAX sta per Asynchronous JavaScript + XML. L'acronimo, ormai, è un po' fuorviante dato che AJAX può essere usato oggigiorno per richiedere al server, oltre che XML, dati in vari formati. La cosa più importante, ed interessante, è che AJAX ci permette di inviare chiamate asincrone al server.
In quest'articolo vedremo come integrare queste due tecnologie per costruire applicazioni performanti di ultima generazione... Web 2.0 appunto.

CONFRONTO TRA JSON E XML

Per apprendere una tecnologia è spesso utile metterla a confronto con una già esistente e conosciuta. A tal proposito modelleremo una realtà classica, ovvero quella rappresentata dalla gestione dell'anagrafica degli impiegati di una società X. Lo faremo prima in XML e poi in JSON. Per semplicità modelleremo l'anagrafica di un impiegato solo attraverso poche proprietà: *Cognome*, *Nome*, *Città*, *Dipartimento*. Il corrispondente in XML potrebbe essere qualcosa del genere:

```
<?xml version='1.0' encoding='UTF-8'?>
<employees>
    <employee>
        <surname>Lacava</surname>
        <name>Alessandro</name>
        <city>Milano</city>
        <department>IT</department>
    </employee>

    <employee>
        <surname>Mautone</surname>
        <name>Giovanna</name>
        <city>Varese</city>
        <department>Affari
            Legali</department>
    </employee>

</employees>
```

La notazione è quella classica della sintassi well formed. Proviamo adesso a rappresentare gli stessi dati attraverso JSON:

```
{
    "employees" :
    [
     {
       "surname" : "Lacava",
         "name" : "Alessandro",
         "city" : "Milano",
         "department" : "IT"
     },
     {
         "surname" : "Mautone",
         "name" : "Giovanna",
         "city" : "Varese",
         "department" : "Affari Legali"
     }
    ]
}
```

Potete già notare la minore "verbosità" di JSON rispetto a XML. Infatti, ci si riferisce spesso a JSON come ad un'alternativa "fat-free" (priva di grasso) di XML. Seppur meno

verboso, è stupefacente come JSON sia tanto espressivo quanto XML. Chiaramente, gli spazi utilizzati all'interno di una stringa in formato JSON sono del tutto arbitrari. Potete quindi indentare i vostri dati come più vi piace. Gli esperti di JavaScript avranno notato che JSON ne è un sottoinsieme. Il codice appena visto, infatti, non è altro che un modo letterale per esprimere oggetti ed array in JavaScript. Vediamo, dunque, di chiarire come sono rappresentati oggetti ed array utilizzando JSON.

DUE PILASTRI FONDAMENTALI

Sostanzialmente JSON viene utilizzato per rappresentare una sequenza di array e/o oggetti. Se, ad esempio, dovessimo rappresentare l'oggetto libro con proprietà: ISBN, Titolo, Autore ed Editore, ci basterebbe scrivere il seguente codice:

```
{"libro" :
        { "ISBN" : 0764579088,
          "titolo" :  "Professional
                 JavaScript for Web Developers",
          "autore" : "Nicholas C. Zakas",
          "editore" : "Wrox"
        }
}
```

In JSON, un oggetto non è altro che un insieme non ordinato di coppie proprietà/valore. Un oggetto inizia con la parentesi graffa aperta e finisce con quella chiusa. Ogni proprietà è seguita dai due punti e dal valore della stessa. Il valore della proprietà può essere uno dei seguenti tipi JavaScript: *stringa, numero, oggetto, array, true, false, null*. Abbiamo rappresentato il campo ISBN come un numero mentre le altre proprietà come stringhe. La stringa libro rappresenta il reference dell'oggetto. In pratica, per costruire lo stesso oggetto in JavaScript avremmo potuto usare il seguente codice:

```
var libro = new Object();
libro.ISBN = 0764579088;
libro.titolo = "Professional JavaScript for Web
                                   Developers";
libro.autore = "Nicholas C. Zakas";
libro.editore = "Wrox";
```

Oppure, utilizzando la notazione letterale:

```
 "ISBN" : 0764579088,
 "titolo" :
   "Professional JavaScript for Web Developers",
 "autore" : "Nicholas C. Zakas",
 "editore" : "Wrox"
};
```

Come potete notare, questa seconda versione assomiglia molto alla sintassi JSON, a riprova del fatto che JSON non è nient'altro che un sottoinsieme di JavaScript. Potreste obiettare che ciò non è vero in quanto la sintassi utilizzata in JSON non è esattamente uguale a quest'ultima vista. Ok, proviamo a confrontare co il seguente codice:

```
var obj = eval(
         {
           "libro" : {
                  "ISBN" : 0764579088,
                  "titolo" : "Professional
                   JavaScript for Web Developers",
                  "autore" : "Nicholas C. Zakas",
                  "editore" : "Wrox"
             }
         }
);
//visualizza il titolo del libro
alert(obj.libro.titolo);
```

In sostanza, abbiamo preso l'oggetto libro, rappresentato utilizzando la sintassi JSON, e l'abbiamo "inglobato" all'interno della funzione eval. Quest'ultima non fa altro che eseguire il codice JavaScript che le viene passato come argomento. Eval si comporta in pratica come un vero e proprio interprete JavaScript. Ad esempio:

```
eval("alert('Ciao Mondo!')");
```

visualizza un alert contenente la stringa Ciao Mondo! Tornando a JSON vediamo, ora, come è possibile rappresentare un array, ad esempio l'array *linguaggi*.

GLI STRUMENTI NECESSARI

Per eseguire gli esempi di quest'articolo avete bisogno di un Web server. Questo perché AJAX comunica proprio con un server per scambiare messaggi HTTP. Quindi, per richiamare la pagina index.htm di esempio non sarà sufficiente fare doppio click su essa. Infatti, dovrete farlo attraverso il Web server. Ad esempio, se avete Apache HTTP Server in ascolto sulla porta di default, dovrete richiamare la pagina utilizzando l'indirizzo: http://localhost/JSON/index.htm. Questo supposto che index.htm si trovi sotto una cartella di nome JSON.

```
{"linguaggi" :
        [
        "Java",
        "C#",
        "JavaScript"
        ]
}
```

In JSON un array è una sequenza ordinata di valori. Tali valori possono essere rappresentati dai tipi JavaScript già descritti in precedenza. Ad esempio, è possibile avere un array di oggetti.

UNA COPPIA VINCENTE!

AJAX è una tecnologia relativamente recente che promette di rivoluzionare il modo di concepire la programmazione Web. Basti pensare che colossi dell'IT quali Google ne fanno già un uso massiccio per le loro applicazioni.
In pratica, utilizzando AJAX si possono inviare richieste asincrone ad un Web server. È possibile, quindi, aggiornare lo stato di un componente della pagina (in base alla risposta ricevuta dal server) senza "refreshare" l'intera pagina. Facciamo subito un semplice esempio. Supponiamo di avere in una pagina due combo box. In una combo è possibile selezionare una regione italiana. Nell'altra, invece, è possibile selezionare la provincia relativa alla regione scelta in precedenza. È chiaro che la combo delle province dipende fortemente dalla combo delle regioni. Ovvero, la combo delle province sarà riempita solo dopo che l'utente seleziona la regione. Vediamo come sarebbe risolto tale problema utilizzando due scenari, quello classico e quello utilizzando AJAX.

Scenario classico:
Senza l'ausilio di AJAX dovremmo ragionare nel seguente modo. Appena l'utente seleziona

Fig. 1: Screenshot dell'applicazione

la regione parte una richiesta al server per recuperare la lista delle province riguardante la regione. Appena tale ricerca finisce, ossia quando riceviamo il response dal server, viene rinfrescata l'intera pagina. Tale refresh aggiorna la combo delle province e ricarica tutti gli altri elementi della pagina, anche se non hanno subito cambiamenti.

Scenario concernente l'utilizzo di AJAX:
Appena l'utente seleziona la regione parte, in modo asincrono e quindi non bloccante, la ricerca delle province associate. Nel momento in cui tale ricerca finisce viene fatto il refresh del solo componente relativo alle province. Ovvero la combo delle province.

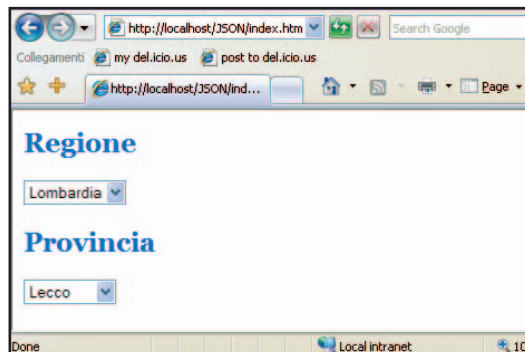
Quello espresso a parole in quest'ultimo caso, è proprio ciò che AJAX ci consente di fare nella pratica.

L'APPLICAZIONE DI ESEMPIO

Implementermo proprio un'applicazione web che data una combobox, consente di selezionare al suo interno il nome di una regione italiana e restituisce l'elenco delle sue province. Il tutto ovviamente senza effettuare il refresh della pagina. Utilizziamo AJAX e JSON per creare le due combo box per la selezione della regione e relativa provincia descritte prima. Il risultato che vogliamo ottenere è visibile in figura 1. I passi da seguire sono i seguenti:

- Creare il codice HTML relativo all'applicazione
- Scrivere il codice JavaScript per la gestione delle chiamate AJAX e l'interpretazione dei messaggi JSON ricevuti.
- Creazione dei messaggi JSON ovvero la rappresentazione di regioni e province.

Per semplicità, i messaggi JSON saranno scritti direttamente in due file di testo. Ovviamente, in un'applicazione per il mondo reale tali messaggi verrebbero creati dinamicamente interrogando, ad esempio, una base di dati. Tuttavia, per lo scopo didattico dell'articolo, i file .txt fanno al caso nostro. Il codice HTML riguardante l'applicazione è molto semplice. Eccone il corpo:

```
<body onload="retrieveRegions();
                    retrieveProvinces();">
```

```
        <form name="provinceItaliane">
                <h2>Regione</h2>
                <div id="regione">
<!-- Qui verra  inserito il risultato della chiamata
AJAX relativa alle regioni-->
                </div>
                <h2>Provincia</h2>
                <div id="provincia">
<!-- Qui verra inserito il risultato della chiamata AJAX
relativa alle province-->
                </div>
        </form>
</body>
```

In pratica, all'evento *load* della pagina sono associate due chiamate ad altrettante funzioni JavaScript. I due *div*, invece, sono dei segnaposto per entrambe le combo box. Esaminiamo le funzioni JavaScript. La funzione *retrieveRegions* ha il seguente codice:

```
//recupera le regioni usando una chiamata AJAX
function retrieveRegions()
{
        //crea l'oggetto XMLHttp
        httpRequestForRegions =
                                createHttpRequest();
        //prepara la richiesta
        //nota: il terzo parametro indica se la
        //richiesta deve essere asincrona o meno
        httpRequestForRegions.open("get",
        "http://localhost/JSON/regioni.txt", true);
        //funzione di callback
httpRequestForRegions.onreadystatechange =
                                displayComboRegions;
        //invia la richiesta
        httpRequestForRegions.send(null);
}
```

La funzione crea un oggetto del tipo *XMLHttpRequest*, utilizzato per fare chiamate asincrone al server. Per fare ciò viene chiamata la funzione *createHttpRequest* che vedremo più avanti. In seguito, viene chiamato il metodo *open* sull'oggetto. Tale metodo accetta tre parametri. Il primo indica il metodo HTTP da utilizzare, nel nostro caso *get*. Il secondo è l'URL a cui inviare la richiesta. Il terzo parametro indica se la richiesta deve essere asincrona. Se è asincrona allora occorre definire una funzione di callback che sarà chiamata all'occorrenza. Nel nostro caso tale funzione è *displayComboRegions*. Infine, viene inviata la richiesta vera e propria tramite il metodo *send*. Tale metodo accetta in ingresso la lista dei parametri da passare al server. Nel nostro caso nessuno. Vediamo, ora, il codice della funzione di callback:

```
//visualizza la combo relativa alle regioni
function displayComboRegions()
{
        //controlla lo stato della risposta
        if(httpRequestForRegions.readyState ==
      4 && httpRequestForRegions.status == 200)
        {
            var jsonObject = eval('(' +
      httpRequestForRegions.responseText + ')');
            var regions =  jsonObject.regioni;
          //combo
          var htmlResponse = "<select
                                name='regions'
            onchange='retrieveProvinces();'>";
              for(i = 0; i < regions.length;
                                          i++)
              {
                        //riempie la combo
                                //delle regioni
                        htmlResponse +=
"<option value='" + regions[i] + "'>" + regions[i]
                                + "</option>";
              }
document.getElementById("regione").innerHTML
                                = htmlResponse;
        }
}
```

La prima cosa che fa è controllare lo stato della risposta e il relativo codice HTTP. Se *readyState* vale 4 e *status* è 200, vuol dire che la richiesta è stata completata e la risposta ha un codice di successo (200). A questo punto entra in gioco JSON. La funzione eval esegue il parsing delle regioni espresse in formato JSON.
Il contenuto del file *regioni.txt*, infatti, non è altro che una stringa in formato JSON:

```
{"regioni" : ["Calabria", "Lazio", "Lombardia"]}
```

Abbiamo indicato solo tre regioni per sempli-

CONSIDERAZIONI SULLA SICUREZZA

La funzione eval è molto potente ma altrettanto pericolosa. Essa esegue, infatti, qualsiasi codice JavaScript le viene passato. Se non siete sicuri della fonte da cui proviene la stringa da parsare, conviene utilizzare delle librerie apposite che effettuano alcuni controlli di sicurezza ed eseguono il parsing solo se riconoscono il messaggio come JSON. Niente paura però, il parsing continua ad essere molto semplice (sempre una riga di codice!). Inoltre, esistono molte librerie per convertire un messaggio JSON in e dai principali linguaggi di programmazione quali: PHP, Java, C# e tanti altri.
Maggiori informazioni le potete trovare sulla pagina ufficiale del progetto: http://www.json.org/

cità. Il codice che segue è abbastanza semplice. Riempie una combo box di nome *regions* con i valori recuperati dal parsing. Successivamente, tale combo viene visualizzata all'interno dell'elemento *regione* (il div) della pagina HTML vista prima.
Vediamo, invece, come funziona il recupero delle province. La funzione *retrieveProvinces* è la seguente:

```
function retrieveProvinces()
{
        httpRequestForProvinces =
                            createHttpRequest();
        httpRequestForProvinces.open("get",
     "http://localhost/JSON/province.txt", true);
httpRequestForProvinces.onreadystatechange =
                        displayComboProvinces;
        httpRequestForProvinces.send(null);
}
```

Come si può vedere è molto simile a *retrieveRegions* se non per il fatto che invia la richiesta al file *province.txt* e ha, come funzione di callback, *displayComboProvinces*. Analizziamo quest'ultima:

```
//visualizza la combo relativa alle province
function displayComboProvinces()
{
        //controlla lo stato della risposta
        if(httpRequestForProvinces.readyState
 == 4 && httpRequestForProvinces.status == 200)
        {
                var jsonObject = eval('(' +
     httpRequestForProvinces.responseText + ')');
                var provinces =
                            jsonObject.province;
                //combo
                var htmlResponse = "<select
                           name='provinces'>";
                for(i = 0; i < provinces.length;
                                          i++)
                {
                //controlla se la  provincia
                //recuperata appartiene
                //alla regione selezionata
                         if(provinces[i].regione
    == document.provinceItaliane.regions.value)
                         {
                //riempie la combo delle province
htmlResponse += "<option value='" +
                     provinces[i].nome + "'>" +
provinces[i].nome +"</option>";
                         }
                }
document.getElementById("provincia").innerHTML
                              = htmlResponse;
        }
}
```

Fig. 2: Lo schema di funzionamento della sintassi Json

L'unica cosa da aggiungere rispetto ai commenti fatti per *displayComboProvinces* è il controllo effettuato per vedere se la provincia "parsata" appartiene alla regione scelta nella combo *document.provinceItaliane.regions*. Solo in tal caso, infatti, la provincia va aggiunta alla combo relativa. Il file province.txt è così strutturato:

```
{"province" :
        [
         {
                "nome" : "Catanzaro",
                "regione" : "Calabria"
        },
        {
                "nome" : "Cosenza",
                "regione" : "Calabria"
        },
        {
                "nome" : "Crotone",
                "regione" : "Calabria"
        },
        {
                "nome" : "Reggio Calabria",
                "regione" : "Calabria"
        },
        {
                "nome" : "Vibo Valentia",
                "regione" : "Calabria"
        },
        {
                "nome" : "Frosinone",
                "regione" : "Lazio"
        },
```

```
                                        [...]
```

In pratica è una stringa JSON che rappresenta l'array *province*. Gli elementi di tale array sono oggetti che hanno due sole proprietà: *nome* e *regione*. Quindi, una volta fatto il parsing utilizzando il metodo *eval*:

```
var jsonObject = eval('(' +
        httpRequestForProvinces.responseText + ')');
```

ed ottenuto un reference a tale array:

```
var provinces = jsonObject.province;
```

è possibile accedere ad un oggetto dell'array e in particolare, ad esempio, alla proprietà *nome* tramite:

```
provinces[i].nome
```

dove i è l'indice dell'array.

OGGETTO XMLHTTPREQUEST

Esaminiamo ora la funzione *createHttp Request* utilizzata in precedenza. Ecco il codice:

```
// Crea un oggetto di tipo XMLHttpRequest
function createHttpRequest()
{
        if (typeof XMLHttpRequest !=
                "undefined") //Il browser non è IE
        {
                return new XMLHttpRequest();
        }
        else if (window.ActiveXObject) // è IE
        {
                var sVersions = [
                        "MSXML2.XMLHttp.5.0",
"MSXML2.XMLHttp.4.0","MSXML2.XMLHttp.3.0",
"MSXML2.XMLHttp","Microsoft.XMLHttp"
                ];
                // Cerca di istanziare la
                // versione più recente.
                // Se non è disponibile prova
                // via via con quelle più datate
                for (var i = 0; i
                        <sVersions.length; i++)
                {
                        try
                        {
                                var ret =
                new ActiveXObject(sVersions[i]);
                                return ret;
```

```
                        }
                        catch (oException)
                        {
                        // Non fa niente.
                        // Va avanti tentando
                        // con le altre versioni
                        }
                }
        }
        //Se arriva qui allora
        // l'oggetto XMLHttpRequest non
        // è disponibile per il browser in uso
        alert("Il browser in uso è obsoleto.
                Aggiornarlo con uno più recente");
}
```

La necessità di utilizzare questa funzione sta nel fatto che Firefox e Opera hanno un modo diverso rispetto ad Internet Explorer di istanziare l'oggetto utilizzato per inviare chiamate AJAX. Il metodo controlla se l'oggetto *XMLHttpRequest* è direttamente disponibile e in tal caso lo istanzia. Tale approccio è utilizzato dai browser diversi da IE. Se l'oggetto non è direttamente disponibile allora si tratta di IE e, in tal caso, l'oggetto in questione è disponibile attraverso un *ActiveX*. Il problema è che vi sono diverse versioni. Il ciclo *for* cerca di istanziare l'oggetto dal più recente al più datato catturando le eventuali eccezioni lanciate per la non disponibilità dell'oggetto. Ovviamente, se arriva alla fine del ciclo vuol dire che l'oggetto non è proprio disponibile per il browser in uso e viene notificato un avviso.

CONCLUSIONI

Chi ha già lavorato con AJAX e XML apprezzerà di sicuro la potenza di JSON.
Effettuare, infatti, il parsing di un documento XML in JavaScript non è una cosa banale. Invece, per estrarre le informazioni che ci interessano da un messaggio JSON ci basta utilizzare un singolo metodo: *eval* per l'appunto.
Questo perché JSON non è altro che un sottoinsieme di JavaScript. Ovviamente lo scopo del presente articolo non è quello di screditare XML il quale rimane sempre un mezzo molto potente per la rappresentazione delle informazioni.
La scelta dell'una o dell'altra soluzione deve essere dettata dal buon senso dello sviluppatore che valuterà le soluzioni caso per caso.

Alessandro Lacava

GUIDA PRATICA ALL'USO DI AJAX

SI FA UN GRAN PARLARE DI WEB 2.0. IL PUNTO È CHE ALLA BASE DI QUESTA NUOVA RIVOLUZIONE DELLA RETE CI SONO TECNOLOGIE ORMAI CONSOLIDATE COME AJAX E JAVASCRIPT. ECCO IL CODICE PER PORTARE LE TUE APPLICAZIONI NELLA NUOVA ERA...

Ultimamente si parla molto della tecnologia AJAX (Asynchronous JavaScript and XML), ovvero la tecnologia che sfrutta le capacità dei browser moderni di "capire" l'XML per creare applicazioni web dinamiche.
Purtroppo è infatti invalsa la tendenza a realizzare controlli e librerie AJAX per le varie piattaforme di programmazione Web lato server (ASP.NET, PHP, Java ecc…) per di più nate spesso con l'intento dichiarato di "nascondere" al programmatore (o per dire più simpaticamente "rendere trasparente") il funzionamento della tecnologia sottostante.
La mia opinione è invece che in tal modo si rischia di avere programmatori che usano una tecnologia senza nemmeno conoscerla (un po' come avviene per alcuni programmatori di pagine web che a forza di usare i vari programmi "visuali" si dimenticano l'HTML).
Peggio ancora si rischia, da parte di programmatori web abituati al "lato server", di non comprendere le reali potenzialità di queste tecnologie che non sono la "ciliegina sulla torta", ma invece possono rappresentare il vero e proprio motore di una Web Application.
Se siamo proprio affezionati agli acronimi più in voga parliamo pure di AJAX, io però preferisco definire questo insieme di tecnologie come *Client Based Web Application* per distinguerle dalle applicazioni tradizionali; *Server Based Web Application*.

CLIENT BASED WEB APPLICATION

Quando dobbiamo creare un'applicazione Web che deve interagire con i dati viene naturale pensare di ricorrere ad una tecnologia che funziona lato server (CGI,ASP,ASP.NET, PHP, Java ecc…), naturalmente ognuna di esse ha le sue peculiarità, ma il flusso alla fine è sempre quello mostrato in figura 1.
In pratica il client (browser) si limita a richiedere una determinata pagina, ma il vero lavoro viene svolto dal server che:

- Interpreta i parametri passati dal client con la url (GET) o con una form (POST)
- Compie le azioni associate ai parametri
- Recupera i dati dal database
- Formatta i dati in HTML dinamicamente
- Restituisce i dati già formattati al client

In pratica, cioè, sia il *Back End* sia il *Front End* dell'applicazione sono demandate al server.
Quali siano le limitazioni di questa tecnologia è facile intuirlo : il server è uno, i client sono molti. Quindi, più sono i client, più sono le risorse richieste al server che deve fare tutto il lavoro.
In questi casi, all'aumentare degli utenti della Web Application si può far fronte solo in un modo: aumentare connettività e potenza del server. Negli ultimi anni però i browser sono cambiati radicalmente rispetto alle versioni dei primi anni della storia del Web: il supporto di javascript, la possibilità di interagire con il DOM (Document Object Model) di HTML e, da ultimo, le capacità di accedere a dati XML.
Questa evoluzione del browser ha portato molti sviluppatori a vederlo come un soggetto attivo della Web Application, un vero e proprio *Front End* dell'applicazione.

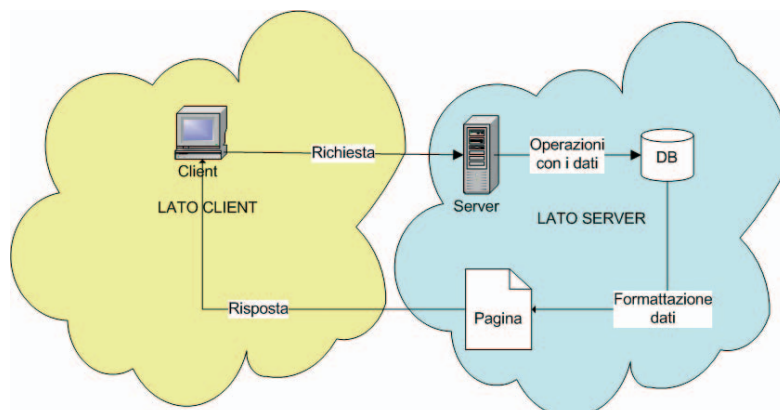



Fig. 1: Flusso di una Web Application Server Based

Ciò ha portato a poter progettare applicazioni che lasciano al server le operazioni di *Back End* e demandano al browser la costruzione dell'interfaccia utente e le altre operazioni client come possiamo vedere in **figura 2**.

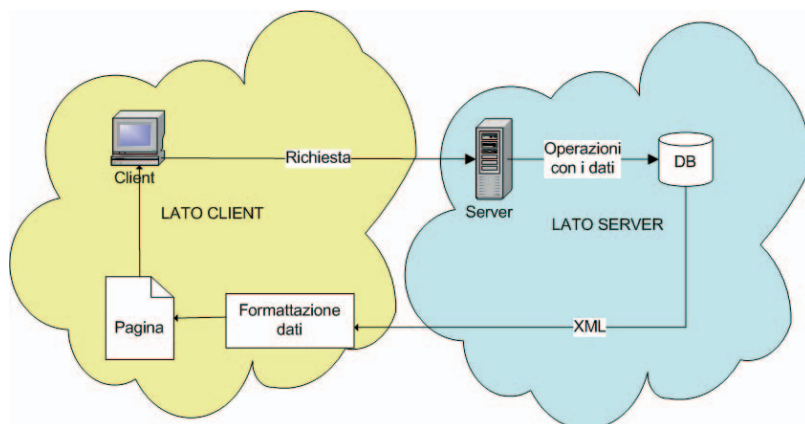



Fig. 2: ***Flusso di una Web Application Client Based***

In un flusso di questo tipo notiamo che:

- Il client richiede al server dei dati
- Il server recupera i dati e li restituisce in formato XML
- Il client elabora questi dati e li utilizza per costruire dinamicamente l'interfaccia utente

I benefici che si ottengono sono di due tipi

- Si alleggerisce notevolmente il carico del server esimendolo dal dover costruire la pagina HTML (e quindi risparmiando banda e risorse server)
- Si possono compiere diverse operazioni anche senza ricaricare l'intera pagina, ma richiedendo solo i dati che servono.

UN ESEMPIO CONCRETO

Ma passiamo subito ad un esempio pratico per capire meglio. In una nostra applicazione dobbiamo implementare un flusso di questo tipo:

1. visualizzare una lista di clienti
2. selezionare un cliente e mostrare le informazioni di dettaglio
3. selezionare gli ordini che il cliente ha effettuato e mostrarli in una tabella

Con un'applicazione *Server Based* opereremmo probabilmente così:

1. La **pagina A** presenta la lista dei clienti in forma di tabella
2. Cliccando sul nome del cliente si apre la **pagina B** con il dettaglio delle informazioni
3. Cliccando su un bottone della pagina B si apre la **pagina C** con la lista degli ordini del cliente

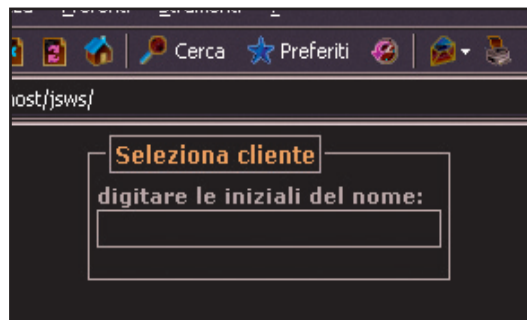



Fig. 3: ***Fase iniziale***

Naturalmente non è sempre necessario dividere il flusso in tre pagine distinte, tuttavia da un passaggio all'altro si avrà sempre il *reload* della pagina. Ma vediamo come potrebbe invece funzionare un'applicazione *Client Based*.
Al caricamento della pagina viene presentata una casella di testo dove l'utente digita le iniziali del nome del cliente (**fig.3**)
Man mano che l'utente digita delle lettere il browser richiede al server i dati dei primi 10 clienti il cui nome inizia per quelle lettere e riempie una lista sottostante (**fig.4**)

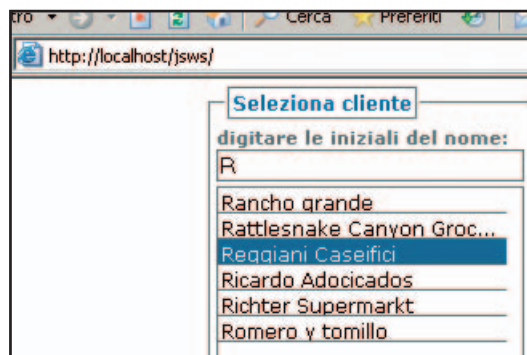



Fig. 4: ***Selezione utente***

Cliccando su un nome della lista, sempre senza ricaricare la pagina, il browser elabora i dati del cliente e presenta un box di dettaglio (**fig. 5**)
Nel dettaglio appare anche un bottone "Ordini" cliccando il quale il browser, sempre in background, richiede al server i dati degli ordini del cliente presentandoli in forma di tabella (**fig. 6**)
Selezionando poi un altro cliente dalla lista l'intera operazione si ripete sempre senza ricaricare la pagina.
Il principio di base è semplice: si caricano una volta sola gli script e l'HTML necessari all'interfaccia utente e poi, volta a volta, che si deve

Fig. 5: ***Dettagli utente***

compiere una determinata operazione si richiedono al server solo i dati necessari e si elaborano di conseguenza. Passiamo quindi a capire il funzionamento pratico della tecnica.

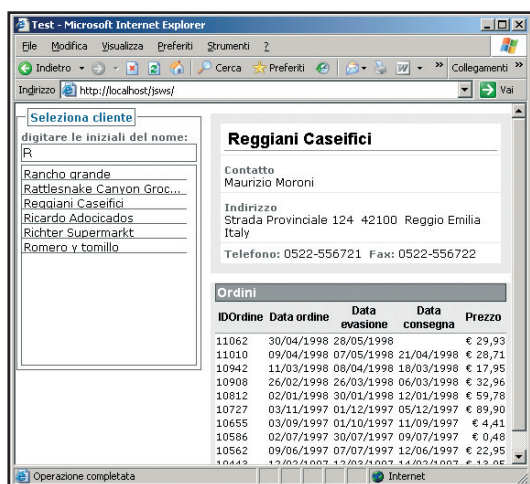



Fig. 6: ***Dettaglio ordini***

LA TECNICA

Le Client Based Web Applications si basano su due pilastri fondamentali:

- Il lato server
- Il lato client

Analizzeremo i due aspetti partendo proprio dall'esempio che abbiamo mostrato.

Il lato server

Dal punto di vista del server l'attività di programmazione è veramente minimale : si tratta di fornire al client i dati nel formato richiesto (generalmente XML): il client richiede, ad esempio, i dati della tabella clienti? Nel nostro sito ci sarà una pagina asp,php o altro che di occupa di leggere i dati dal database e restituirli nel formato richiesto. Ma se XML è il formato di elezione per questo tipo di colloqui, qual è la tecnologia lato server che "parla" in XML? Ovviamente i famosi Web Service. Niente di meglio quindi di un Web Service per fornire i dati al client. Oltretutto, utilizzando un Web Service abbiamo anche il vantaggio di poter riutilizzare le stesse funzioni di accesso ai dati anche con applicazioni desktop. I Web Service possono essere realizzati con varie piattaforme di sviluppo, noi, per il nostro esempio abbiamo scelto di lavorare in .NET con Visual Basic. Nella root del nostro sito quindi creiamo lo "scheletro" del nostro Web Service in un file che chiameremo server.asmx :

```
<%@ WebService Language="VB" Class="Server"
                                          %>
Imports ...
<WebService(Namespace:="http://myservice")> _
Public Class Server
    Inherits System.Web.Services.WebService
    Private ReadOnly Property Database() As DataSet
 ...
    End Property
    Private ReadOnly Property Customers() As
                                        DataTable
...
    End Property
    Private ReadOnly Property Orders() As DataTable
...
    End Property
    <WebMethod()> _
    Public Function GetCustomers(ByVal
                companyName As String) As DataSet
...
    End Function
    <WebMethod()> _
    Public Function GetOrders(ByVal customerID As
                              String) As DataSet
        ...
    End Function
End Class
```

Naturalmente qui abbiamo omesso tutta la logica implementativa vera e propria (che trovate nei sorgenti).
È tutto molto semplice:

- Leggiamo i dati da un **DataSet** (che noi nell'esempio abbiamo salvato **come** file XML, ma che in produzione sarà il risultato di una query al database) nella proprietà Database.
- Nel **DataSet Database** ci sono due tabelle, Customers e Orders, rappresentate dalle rispettive proprietà.
- Forniamo infine due metodi pubblici che saranno il canale di comunicazione con il client; il primo, GetCustomers, restituisce un nuovo **DataSet** con le prime 10 righe dei clienti il cui nome inizia per il valore del parametro **companyName**, **GetOrders** invece restituisce un altro **DataSet** con tutte le righe d'ordine relative

"Programming ASP.NET 2.0 – Core Reference", Dino Esposito (Microsoft Press)
"Professional ASP.NET 2.0", Evjen et al. (Wrox Press)

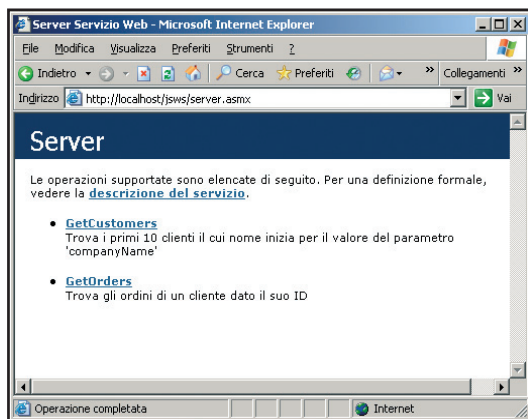



Fig. 7: *Il web service in azione*

ad un cliente attraverso il suo ID.
Da ultimo, nel file web.config della nostra Web Application inseriamo :

```
<configuration>
  <system.web>
```

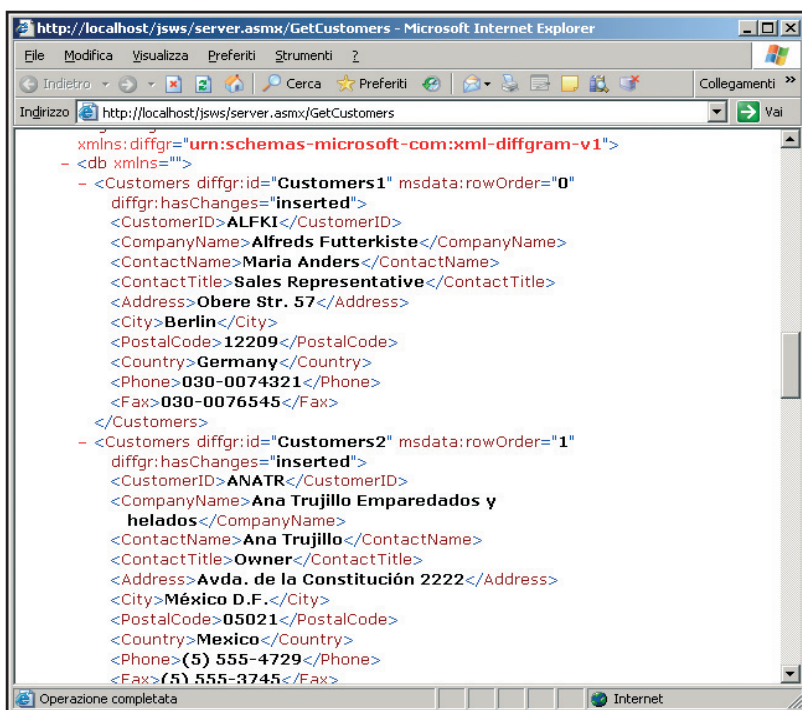



Fig. 8: *Il DataSet prodotto da GetCustomers*

UTILIZZARE WEB SERVICES ESTERNI

Abbiamo visto che nell'esempio trattato nell'articolo si richiamano Web Services presenti nell'ambito dello stesso dominio dove risiede la pagina web chiamante (la URL infatti è relativa non assoluta). C'è una limitazione di sicurezza dell'oggetto XMLHttpRequest che non consente di accedere a risorse esterne al dominio. Qualora fosse necessario accedere a Web Services situati in altri siti dovremmo creare nel nostro sito un Web Service che fa da proxy verso i servizi esterni e da interfaccia verso le chiamate di XMLHttpRequest.

```
    …
    <webServices>
      <protocols>
        <add name="HttpPost"/>
      </protocols>
    </webServices>
  </system.web>
</configuration>
```

questo ci garantisce che possiamo interagire con il Web Service anche con il metodo POST anziché solo con SOAP (che da Javascript sarebbe molto più complicato da maneggiare).
A questo punto abbiamo già finito il nostro lavoro sul lato server: configuriamo, in IIS, una virtual directory corrispondente alla path della nostra applicazione e la chiamiamo, ad esempio, *jsws* (javascript + web service).
A questo punto siamo in grado di accedere al servizio con il browser all'indirizzo : http://localhost/jsws/server.asmx, se tutto è andato bene dovremmo vedere l'interfaccia classica dei Web Service .NET che presenta la lista delle funzioni (**fig. 7**). Se provassimo poi a richiamare la funzione GetCustomers il risultato sarà la rappresentazione del DataSet in formato XML che osserviamo in **figura 8**. A questo punto si tratta di vedere come richiamare ed utilizzare questi dati con il client e quindi con javascript.

Il lato client

Il fulcro della tecnologia lato client è rappresentato da due oggetti:

- **XMLHttpRequest** che si occupa di recuperare i dati da una URL con i metodi GET o POST
- **XMLDOMParser** che si occupa di leggere un documento XML

La nota dolente è che i vari browser moderni (parliamo principalmente di IE e Firefox) benché supportino tali oggetti lo fanno (poteva essere diversamente?) in modo differente: possiamo dire che mentre Firefox è più aderente agli standards, ma molto più difficile da utilizzare, IE estende gli standards con funzioni proprietarie, che d'altro canto semplificano notevolmente la vita del programmatore. La soluzione sta nello sviluppare una libreria Javascript (ovvero un semplice file .js da includere nelle nostre pagine HTML) che fornisca un oggetto intermedio che consenta di utilizzare XMLHttp Request e il parser XML senza preoccuparsi troppo del browser di destinazione.

LA LIBRERIA DI SUPPORTO

Noi, per ovviare al problema, abbiamo realizzato la libreria xml.js (che trovate nei sorgenti di esempio) che in pratica riproduce anche su Firefox i metodi

che troviamo nel parser XML di Microsoft.
All'interno della libreria l'oggetto che rappresenta l'accesso alle funzioni XML è *jsXML* dichiarato a livello globale come:

```
var jsXML = new Object();
```

jsXML dispone di varie funzioni per l'utilizzo di **XMLHttpRequest** e **XMLDOMParser**, vediamo come crea un oggetto **XMLHttpRequest**:

```
jsXML.getXMLHttpRequest = function(){
        //per Firefox
        if (typeof(XMLHttpRequest) != 'undefined')
                return new XMLHttpRequest();
        //per IE
else return new ActiveXObject ("Msxml2.XMLHTTP");
}
```

L'oggetto XMLHttpRequest può caricare i dati da una URL in maniera asincrona o sincrona, utilizzando GET o POST, le operazioni sincrone sono compiute da :

```
jsXML.sendSync = function(url,body){
        var req = this.getXMLHttpRequest();
method=(arguments[2])?arguments[2]:"POST";
        req.open(method, url, false);
        if (method=="POST") req.setRequestHeader
          ("Content-Type", "application/x-www-form-
                                   urlencoded");
        req.send(body);
        return req.responseText;
}
```

Dove il metodo è di default il POST (salvo che non venga specificato diversamente da un terzo parametro) e i parametri sono: url (l'indirizzo relativo della risorsa) e body (i parametri nel formato urlencoded: *parametro1=xxx¶metro&=yyy*). Come valore di ritorno restituisce la risposta del server. Ad esempio, per richiamare i dati dal nostro Web Service in modo sincrono si può utilizzare la funzione in questo modo:

```
var datiClienti =
 jsXML.sendSync("server.asmx/GetCustomers","compa
                              nyName=abc");
```

Le operazioni asincrone di recupero dei dati sono invece rappresentate da :

```
jsXML.sendAsync = function
        (url,body,method,action,onfault,onwait) {
        var req = this.getXMLHttpRequest();
        var onreadystate = function (){
                if (req.readyState == 4) {
                        if (req.status == 200) {
if(action!=null) action(req.responseText,req);
```

```
                        }
                        else {
if(onfault!=null) onfault(req.statusText);
                        }
                }
                else {
                        if(onwait!=null)
                         onwait(req.readyState);
                }
        }
        req.onreadystatechange = onreadystate;
        req.open(method, url, true);
        if (method=="POST") req.setRequestHeader
          ("Content-Type", "application/x-www-form-
                                   urlencoded");
        req.send(body);
}
```

Qui i parametri sono rappresentati da : *url* e *body* (che avevamo visto in precedenza), *method* (GET o POST). Ci sono poi tre parametri che rappresentano le funzioni di *callback* (opzionali) che vengono attivate rispettivamente:

- **action** - alla ricezione dei dati (riceve il testo della risposta e l'oggetto XMLHttpRequest che l'ha richiesta)
- **onfault** - al verificarsi di un errore (riceve il messaggio di errore)
- **onwait** – nell'attesa dei dati (riceve il codice di stato)

Nel corpo della funzione troviamo poi la sub-funzione onreadystate che gestisce l'evento onreadystatechange dell'oggetto XMLHttpRequest e attiva i vari callback se impostati.
Ad esempio, per richiamare i dati dal nostro Web Service in modo asincrono si può utilizzare la funzione in questo modo:

```
function caricaDati(){
var arrivoDati = function (responseText){
                //interpretazione dati
}
jsXML.sendAsync
 ("server.asmx/GetCustomers","companyName=abc","
```

XML CON JAVASCRIPT

La libreria di funzioni javascript per la gestione degli oggetti XML che proponiamo nell'articolo e alleghiamo nel codice sorgente funziona con IE e Firefox , tuttavia il programmatore che volesse approfondire la gestione di XML con javascript (e consigliamo comunque di farlo) può trovare utile documentazione in : http://msdn.microsoft.com/xml per il mondo Microsoft e http://developer.mozilla.org/en/docs/XML_in_Mozilla per Firefox/Mozilla

```
                                POST",arrivoDati)
}
```

La fase di interpretazione dei dati in arrivo viene cioè demandata ad un'altra funzione che viene passata come parametro di *callback*. La libreria xml.js prevede inoltre altre funzioni di gestione dell'XMLHttpRequest per semplificare ulteriormente l'utilizzo: **jsXML.sendAsyncGet()** per inviare la richiesta asincrona con GET e **jsXML.send AsyncPost()** per inviare la richiesta asincrona con POST. Risolto il problema del recupero dei dati dal Web Service (o da qualsiasi altra fonte) resta quindi quello di interpretarli come XML.
L'oggetto jsXML per questo prevede due metodi cross-browser per instanziare il parser XML caricare i dati da una stringa XML :

```
jsXML.createDOMDocument =
        function(strNamespaceURI, strRootTagName) {
    var objDOM = null;
    if (isMoz) {
        objDOM =
  document.implementation.createDocument(strNames
                    paceURI, strRootTagName, null);
        objDOM.addEventListener("load",
                         _Document_onload, false);
    }
  else{
        objDOM = new ActiveXObject
                         ("Msxml2.DOMDocument")
    }
    return objDOM;
}
```

che crea il parser a seconda dei vari browser, e

```
jsXML.parseXMLDocument = function (xmlText,ns) {
            var space = (ns)?ns:'';
            var doc =
               jsXML.createDOMDocument(space,'root');
            doc.loadXML (xmlText);
            return doc;
}
```

che carica il documento direttamente da una stringa XML. A questo punto per richiamare i dati dal nostro Web Service in modo sincrono è sufficiente utilizzare:

```
var datiClienti =
 jsXML.sendSync("server.asmx/GetCustomers","compa
                                  nyName=abc");
var xmlClienti =
              jsXML.parseXMLDocument(datiClienti);
```

dove *xmlClienti* è l'oggetto XMLDocument pronto per essere letto. Per la lettura dell'oggetto (selezione dei nodi, recupero valori ecc…) si utilizzano metodi e proprietà del parser Microsoft (visto che la libreria li rende disponibili anche per Firefox), quindi, ad esempio, per ottenere con XPath la lista dei nodi Customers del documento XML proveniente dal Web Service sarà sufficiente :

```
var nodes = xmlClienti.selectNodes("//Customers")
```

INTERFACCIA UTENTE

Come avrete capito, l'interfaccia utente viene costruita da un mix di HTML e CSS manipolati con javascript nel file: *index.html*.
Come esempio vediamo il flusso di operazioni che viene innescato quando l'utente digita alcuni caratteri nella casella di testo del nome del cliente (alcune funzioni non sono riportate per brevità ma possono essere analizzate nei sorgenti)
Innanzitutto includiamo nel file html la libreria di funzioni *xml.js* con :

```
<script language="javascript" type="text/javascript"
                         src="lib/xml.js"></script>
```

Nel codice HTML definiamo l'elemento <input> che rappresenta la casella di testo

```
<input type="text" id="CompanyName"/>
```

e l'elemento <div>, inizialmente vuoto, che ospiterà i risultati della ricerca comunicati dal Web Service

```
<div id="RisultatiRicerca"
                   style="display:none"></div>
```

entrambi sono qualificati da un ID.
Nella sezione <script> della pagina impostiamo la funzione che fa partire la richiesta al Web Service:

```
          var occupato = false;
         function richiediDatiClienti(){
                  if (occupato) return ;
                  var datiArrivati = function
                                   (responseText){
                           var docClienti =
              jsXML.parseXMLDocument(responseText);
                           elaboraDati (docClienti)
                           occupato = false;
                  }
                  var errore = function
                                     (statusText){
                                  …
                  }
                  occupato=true;
                  var serviceUrl =
```

L'AUTORE

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop. Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica francesco@smelzo.it

```
"Server.asmx/GetCustomers"
var serviceParameters =
"companyName=" + txtCompanyName.value;
jsXML.sendAsyncPost(serviceUrl,serviceParameters,da
tiArrivati,errore,null);
}
```

dove la richiesta parte con *jsXML.sendAsyncPost* che, all'arrivo dei dati, fa partire la sub-funzione *datiArrivati* la quale, a sua volta, richiama la funzione elaboratati passandogli come parametro il documento XML ottenuto dal Web Service. La funzione *elaboraDati* cancella i risultati precedenti e si occupa di scorrere i nodi *Customers* del documento XML :

```
function elaboraDati(docClienti){
var nodes =
docClienti.selectNodes("//Customers");
//trova con XPath tutti i nodi
<Customers> del documento XML
divRisultatiRicerca.innerHTML =
""; //cancella i risultati preesistenti;
for(var i=0;i<nodes.length;i++){
var nodoCliente =
nodes[i];
inserisciRigaCliente
(nodoCliente);
}
show(divRisultatiRicerca);
}
```

Man mano che il ciclo *for* scorre i nodi <Customers> passa il singolo nodo alla funzione inserisciRiga Cliente che si occupa di costruire dinamicamente la riga da inserire nel pannello risultati:

```
function inserisciRigaCliente(nodoCliente){
var divRigaCliente =
document.createElement("div"); //crea un nuovo
<div></div>
divRigaCliente.innerHTML =
divRigaCliente.title =
getElementValue(nodoCliente,"CompanyName");
divRigaCliente.nodo = nodoCliente
//proprietà aggiunta per estensione
divRigaCliente.className =
"RigaCliente";
divRigaCliente.onmouseover
=function (){ this.className='RigaClienteOver'}
divRigaCliente.onmouseout
=function (){ this.className='RigaCliente'}
divRigaCliente.onclick =
selezioneCliente;
divRisultatiRicerca.appendChild(divRigaCliente);
//aggiunge il div al pannello risultati
}
```

la singola riga viene realizzata costruendo, con il DOM HTML, un nuovo elemento <div> che verrà aggiunto al contenitore RisultatiRicerca.
Da notare che il nodo XML originario resta associato al <div> creato semplicemente aggiungendo la proprietà "nodo" all'elemento per assegnazione in :

```
divRigaCliente.nodo = nodoCliente
```

questa è una delle caratteristiche più utili di javascript; la possibilità di aggiungere nuove proprietà a oggetti semplicemente dichiarandole. Aver associato il nodo XML alla riga dei dati ci tornerà infatti utile per gestire l'evento *onclick* che viene assegnato alla funzione *selezioneCliente* di cui riportiamo alcuni passi:

```
function selezioneCliente(){
…
var elementiCliente =
this.nodo.selectNodes("*");
for(var
i=0;i<elementiCliente.length;i++){
…
}
…
}
```

solo per notare che i dati del nodo XML originario Customers sono richiamabili, nel contesto dell'elemento <div> su cui l'utente fa click, con **this.nodo** dove *this* rappresenta l'elemento stesso. Tutto il resto dell'applicazione si basa sugli stessi concetti : richiedere ed elaborare i dati dal Web Service e costruire dinamicamente l'HTML di risposta.

CONCLUSIONI

L'esempio proposto non rappresenta che una minima parte delle possibilità offerte da questa tecnologia. In particolare non abbiamo parlato dell'aggiornamento dei dati del database attraverso il colloquio Javascript/Web Service, delle questioni di sicurezza e autenticazione ecc... Tuttavia, in questa sede, era interessante vedere la tecnologia come possibile valida alternativa alle applicazioni Web tradizionali.
È vero che l'impegno iniziale nell'apprendimento può essere reso arduo dal mix di competenze in gioco (javascript, HTML, XML, CSS, Web Services) dove invece spesso si ha una netta "divisione dei compiti" tra programmatori e web designer, tuttavia crediamo che, alla fine, i risultati ripaghino ampiamente questo sforzo e rendano il programmatore Web più "completo".

Francesco Smelzo

DOTNETNUKE FACILE E PERSONALIZZABILE

CREIAMO UN PORTALE IN POCHI MINUTI, MODIFICHIAMOLO ATTRAVERSO VISUAL STUDIO ED INFINE ESTENDIAMONE LE FUNZIONALITÀ ATTRAVERSO L'USO DI MODULI. AL TERMINE IL NOSTRO SITO SARÀ DOTATO DI UN BELLISSIMO MOTORE DI RICERCA PER CONTENUTI

DotNetNuke è un CMS (Content Management System) open source scritto in VB.Net. La prima versione è nata da un'applicazione che veniva fornita da Microsoft come esempio di programmazione per il nuovo framework .Net: *IBuySpy Portal*. DotNetNuke consente di sviluppare in modo rapido dei portali web multilingua senza scrivere una riga di codice. La forza di DotNetNuke risiede principalmente nella sua "estendibilità" dal momento che permette di aggiungere dei moduli realizzati da terze parti o di crearne dei propri. E', inoltre possibile, grazie alla separazione tra design e contenuto, realizzare delle skin che consentono di personalizzare l'aspetto grafico del sito realizzato.DotNetNuke è scaricabile dal sito http://www.dotnetnuke.com/

PERSONALIZZARE UN SITO DOTNETNUKE

Chi utilizza Visual Web Developer o Visual Studio 2005 può scaricare dal sito lo "*Starter Kit*", ovvero un framework che consente di integrare DotNetNuke nell'ambiente di sviluppo di Microsoft. In questo modo sarà facile personalizzare un sito realizzato con DNN per le proprie esigenze. Installato lo Starter Kit, in Visual Web Developer compare un nuovo tipo di template chiamato "DotNetNuke Application Framework" che ci consente di realizzare applicazioni web basate sul framework di DotNetNuke.
Clicchiamo su "File", "New Web Site..."
Poiché DotNetNuke è scritto in VB.Net, selezioniamo "Visual Basic" come linguaggio.
Cliccando su "Ok", viene generato lo scheletro dell'applicazione e visualizzato un file contenente alcune istruzioni su come procedere nella configurazione del nostro web site.
Una volta creato il progetto, la prima cosa da fare è creare un database vuoto. Possiamo farlo utilizzando "Sql Server Management Studio"
Oppure tramite una query

```
USE [master]
GO
CREATE DATABASE [CSETest] ON  PRIMARY
( NAME = N'CSETest', FILENAME =
    N'C:\Programmi\Microsoft SQL
    Server\MSSQL.1\MSSQL\DATA\CSETest.mdf' , SIZE =
    20480KB , MAXSIZE = 102400KB , FILEGROWTH =
    10240KB )
 LOG ON
( NAME = N'CSETestLog', FILENAME =
    N'C:\Programmi\Microsoft SQL
    Server\MSSQL.1\MSSQL\DATA\CSETest.ldf' , SIZE =
    5120KB , MAXSIZE = 15360KB , FILEGROWTH =
    1024KB )
 COLLATE Latin1_General_CI_AS
GO
```

Creato il database, occorre editare il file di configurazione. Innanzitutto, cerchiamo nella root un file chiamato ***release.config*** e rinominiamolo "***web.config***".
Apriamolo e modifichiamo le informazioni per l'accesso al database.
Nella sezione <***connectionstrings***> aggiungiamo

```
<add
name="SiteSqlServer"
connectionString="Data
    Source=CRI\MYSQLEXPRESS;Initial
    Catalog=CSETest;Integrated Security=True;"
providerName="System.Data.SqlClient" />
```

Mentre in <***appsettings***>

```
<add
key="SiteSqlServer"
value="Data Source=CRI\MYSQLEXPRESS;Initial
    Catalog=CSETest;Integrated Security=True;"/>
```

e togliamo le stringhe di connessione inserite di default.
A questo punto non ci resta che premere "***Ctrl+F5***" per far partire il setup. Se non abbiamo commesso errori di configurazione, dovrebbe

Conoscenze richieste
SQL Server 2005, .Net Framework 2.0

Software
SQL Server 2005, Visual Studio 2005, DotNetNuke Starter Kit

Impegno
Tempo di realizzazione

comparire una finestra simile a quella in figura.

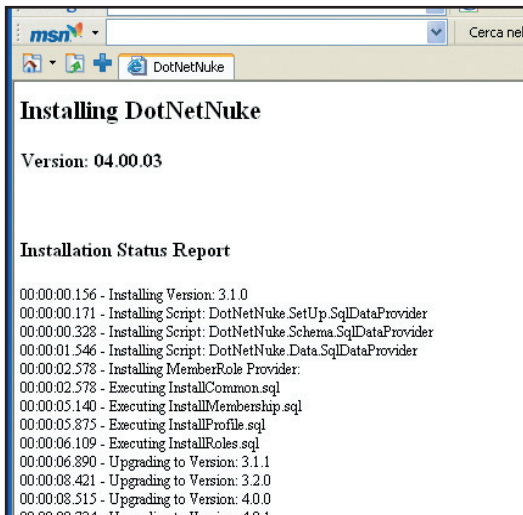



Fig. 6: Installazione completata

Se tutto è andato a buon fine, possiamo cambiare il file di avvio dell'applicazione. Clicchiamo con il tasto destro sul file ***Default.aspx*** presente nella root del sito e selezioniamo "***Set As Start Page***". Premiamo "***Ctrl+F5***" per eseguire l'applicazione e visualizzare il nostro portale.
A questo punto abbiamo già a disposizione un CMS molto potente. Per renderci conto delle funzionalità è sufficiente eseguire il login come amministratore (user e password sono di default ***admin***).

I MODULI DI DOTNETNUKE

DotNetNuke fornisce all'utente numerose funzionalità di base facendone un prodotto molto interessante per un'utenza media ma il vero punto di forza del CMS è la possibilità di realizzare dei ***moduli*** personalizzati che rendono praticamente illimitate le potenzialità di applicazione. Per aggiungere un nuovo modulo, è sufficiente cliccare con il tasto destro del mouse sulla root del progetto e selezionare "*Add New Item...*"
Dalla maschera successiva selezioniamo *DotNetNukeModule*, assegniamo un nome al modulo (nel nostro esempio sarà ***CSE***) e clicchiamo su *Add*. A questo punto, Visual Studio crea per noi il nuovo modulo. È necessario adesso rinominare le due cartelle "*ModuleName*" create da Visual Studio e dargli il nome che abbiamo scelto per il modulo. Per farlo, è sufficiente cliccare con il tasto destro del mouse sulle cartelle e selezionare "*Rename*"
Una volta rinominate le cartelle, dobbiamo apportare delle modifiche al database per registrare il nuovo modulo in DotNetNuke.

Per farlo, apriamo il file chiamato CSE.SqlDataProvider che contiene uno script SQL da eseguire sul database. DotNetNuke fornisce uno strumento per eseguire lo script dall'interfaccia del CMS in modo abbastanza semplice. Per farlo, occorre collegarsi come host al sito DNN che stiamo modificando (la user e la password sono di default "host"). Una volta collegati, selezioniamo dal menu a tendina: *Host->SQL*. Copiamo lo script ed incolliamolo nel box di testo. Selezioniamo "*Run as Script*" e clicchiamo quindi su "*Execute*". Eseguiamo la stessa operazione con lo script *01.00.00.SqlDataProvider* il quale contiene il codice per creare le tabelle e le stored procedures necessarie al funzionamento del modulo. A questo punto siamo in grado di aggiungere il nuovo modulo al sito creato.
Andiamo nella pagina in cui desideriamo utilizzare il modulo, selezioniamo il modulo dal menu a tendina accanto a "*New Module*", scegliamo la posizione in cui vogliamo visualizzarlo e clicchiamo su "*Add*". Il nostro primo modulo è pronto!

REALIZZIAMO UN "CROSS-SEARCH ENGINE"

Per il nostro esempio realizzeremo un modulo che permette all'utente di memorizzare (e poi ricercare) in un database degli articoli salvando titolo, testo e keywords. Il modulo consentirà di individuare all'interno del database gli articoli di nostro interesse utilizzando una semplice funzione di filtro per keyword. Inserita la parola chiave, comparirà un elenco degli articoli ad essa correlati. Cliccando su ognuno di essi, ne verrà visualizzato il dettaglio e accanto a questo una lista ulteriore di articoli con i quali il nostro articolo "di partenza" condivide una o più parole chiave. Partendo dal modulo base illustrato nel paragrafo precedente, esaminiamo le procedure necessarie per la ersonalizzazione. Chiameremo il nuovo modulo CSE (cross-search engine).
Creiamo la pagina in cui andremo ad inserire il modulo. In alto a sinistra, sotto "*Page Functions*", troviamo un link con scritto "*Add*"; clicchiamoci sopra e compiliamo i campi necessari per la creazione di una nuova pagina; clicchiamo quindi su "*Update*". Aggiungiamo poi il modulo alla pagina, come visto in precedenza.
Se non abbiamo commesso errori, effettuando il logout, dovremmo trovare una nuova pagina con il modulo appena installato.
Nella costruzione del modulo vedremo in sequenza i tre livelli dell'architettura di DotNetNuke; DAL (Data Access Layer), BLL (Business Logic Layer) e UI (User Interface), il

DOVE TROVO LO STARTER KIT?
E' possibile scaricarlo all'indirizzo: http://www.dotnetnuke.com/tabid/125/default.aspx Una volta scaricato il file, è sufficiente eseguirlo per far partire l'installazione.

FORMATI DI DATABASE
Con DotNetNuke possiamo anche utilizzare dei file mdf, il nuovo formato introdotto con SQL Server 2005. Per farlo basta cliccare con il tasto destro del mouse sulla cartella App_Data, selezionare "Add New Item" -> "SQL Database" e dare un nome al database. In questo caso nel web.config è necessario cambiare soltanto il nome del datasource.

livello di presentazione. Il DAL fornisce l'accesso ai dati: modificheremo la tabella del modulo ed i file "*SqlData Provider.vb*" e "*DataProvider.vb*".
Il BLL fornisce il collegamento tra i dati e l'interfaccia: i file che andremo a modificare sono: "*CSEController.vb*" e "*CSEInfo.vb*".
La UI fornisce l'interfaccia grafica: modificheremo i file della cartella "*DesktopModules/CSE*"

L'ACCESSO AI DATI

Partiamo dal livello più basso, quello dell'accesso ai dati. Come prima cosa, modifichiamo la tabella "*YourCompany_CSE*" aggiungendo i campi che ci serviranno per realizzare il modulo "cross-search". La cosa più semplice da fare è eliminare la tabella e ricrearla con i campi; possiamo farlo in diversi modi. Utilizzando "SQL Server Management Studio" in modalità visuale.
Oppure con una query

```
USE [CSETest]
GO
DROP TABLE [dbo].[YourCompany_CSE]
GO
CREATE TABLE [dbo].[YourCompany_CSE](
            [ModuleID] [int] NOT NULL,
            [ItemID] [int] IDENTITY(1,1) NOT NULL,
            [Title] [varchar](100) COLLATE
                          Latin1_General_CI_AS NOT NULL,
            [Keywords] [varchar](255) COLLATE
                          Latin1_General_CI_AS NOT NULL,
            [Content] [ntext] COLLATE
                          Latin1_General_CI_AS NOT NULL,
            [CreatedDate] [datetime] NOT NULL,
 CONSTRAINT [PK_YourCompany_CSE] PRIMARY KEY
                                        CLUSTERED
(
            [ItemID] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Che possiamo eseguire nel query analyzer oppure sfruttando la funzionalità di DotNetNuke vista prima. A questo punto, dobbiamo creare le stored procedure per poter gestire la nuova struttura della tabella.

```
CREATE PROCEDURE {databaseOwner}
   [{objectQualifier}YourCompany_CSE_Delete]
(
@ModuleID int,
@ItemID int
)
AS
DELETE FROM YourCompany_CSE
WHERE (ModuleID = @ModuleID AND ItemID =
                                      @ItemID)
CREATE PROCEDURE {databaseOwner}
   [{objectQualifier}YourCompany_CSE_GetAll]
(
@ModuleID int
)
AS
SELECT [ModuleID]
      ,[ItemID]
      ,[Title]
      ,[Keywords]
      ,[Content]
      ,[CreatedDate]
FROM {objectQualifier}YourCompany_CSE
WHERE (ModuleID = @ModuleID)
order by CreatedDate DESC
```

Per eseguirlo, entriamo come host e sfruttiamo la solita funzione *Host->SQL* (con l'opzione "Run as Script" selezionata). Passiamo al file "*SqlDataProvider.vb*" che dobbiamo modificare per fargli utilizzare le stored procedure appena create. La Region da modificare è "*Public Methods*"

```
#Region "Public Methods"
Public Overrides Function GetCSEs(ByVal ModuleId
                        As Integer) As IDataReader
            Return
    CType(SqlHelper.ExecuteReader(ConnectionString,
     GetFullyQualifiedName("CSE_GetAll"), ModuleId),
                                       IDataReader)
End Function
Public Overrides Function GetCSEsFilter(ByVal
    ModuleId As Integer, ByVal Keyword As String) As
                                        IDataReader
            Return
    CType(SqlHelper.ExecuteReader(ConnectionString,
   GetFullyQualifiedName("CSE_GetFilter"), ModuleId,
                              Keyword), IDataReader)
End Function
[...]
#End Region
```

Per brevità abbiamo riportato un paio di funzioni, il resto potete reperirlo nel codice che trovate nel cd allegato alla rivista

LA BUSINESS LOGIC

Nella cartella CSE all'interno di *App_code* troviamo un file chiamato *CSEInfo.vb* che contiene il codice per l'accesso alle proprietà della tabella CSE. Apriamolo e modifichiamolo nel seguente modo:

```
Imports System
Imports System.Configuration
Imports System.Data
Namespace YourCompany.Modules.CSE
    Public Class CSEInfo
        Private _ModuleId As Integer
        Private _ItemId As Integer
        Private _Title As String
        Private _Keywords As String
        Private _Content As String
        Private _CreatedDate As
                DateTime
        ' initialization
        Public Sub New()
        End Sub
        ' public properties
        Public Property ModuleId() As
                    Integer
            Get
                Return
                  _ModuleId
            End Get
            Set(ByVal Value As
                  Integer)
                _ModuleId =
                    Value
            End Set
        End Property

        Public Property ItemId() As
                    Integer
            Get
                Return
                  _ItemId
            End Get
            Set(ByVal Value As
                  Integer)
                _ItemId =
                    Value
            End Set
        End Property
        Public Property Title() As String
            Get
                Return _Title
            End Get
            Set(ByVal Value As
                String)
                _Title =
                Value
            End Set
        End Property
      Public Property Keywords() As String
        Get
            Return _Keywords
        End Get
        Set(ByVal Value As String)
            _Keywords = Value
        End Set
```

```
                End Property
              Public Property Content() As String
              Get
                  Return _Content
              End Get
              Set(ByVal Value As String)
                      _Content = Value
              End Set
        End Property
        Public Property CreatedDate() As DateTime
        Get
            Return _CreatedDate
        End Get
        Set(ByVal Value As DateTime)
                _CreatedDate = Value
        End Set
        End Property
        End Class
End Namespace
```

A questo punto se compiliamo il progetto, compariranno degli errori nel file CSEController.vb. Niente paura, era tutto previsto...
Occorre modificare due file: il primo è DataProvider.vb, che contiene i metodi astratti di cui dovremo fare override, il secondo è proprio CSEController.vb. Apriamo DataProvider.vb e sostituiamo la Region "Abstract methods" con il seguente codice che trovate all'interno del CD
Adesso abbiamo i metodi astratti corretti e possiamo modificare la classe Controller.
Sostituiamo il codice della Region "Public Methods" con i metodi approropriati. Anche di questi per brevità riportiamo un estratto, li trovate interamente nel cd allegato.

```
#Region "Public Methods"
Public Function GetCSEs(ByVal ModuleId As Integer)
                    As List(Of CSEInfo)
        Return CBO.FillCollection(
            Of CSEInfo)(DataProvider.Instance().
                    GetCSEs(ModuleId))
End Function
```

L'INTERFACCIA UTENTE

Possiamo trovare i file relativi alla User Interface nella cartella DesktopModules\ CSE.
EditCSE.ascx è il file che ci consentirà di aggiungere e modificare degli elementi del nostro database, mentre lo User Control ViewCSE.ascx, fornirà l'output per il nostro modulo.
Vediamo prima il file di Edit: la form è composta da due TextBox, una per il title ed una per le Keywords ed componente Textarea di DotNetNuke per il corpo dell'articolo.

ERRORI TIPICI

Se quando inseriamo il modulo nelle nostre pagine otteniamo l'errore "MinMax persistance type of cookie requires a ModuleId", occorre spostare i file presenti in "DesktopModules\CSE\CSE" nella cartella "DesktopModules\CSE\"

Utilizzeremo la form per le operazioni di Insert\Edit\Delete. Il file responsabile delle suddette operazioni è EditCSE.ascx.vb

```
[...]
Private Sub Page_Load(ByVal
            sender As System.Object, ByVal e As
            System.EventArgs) Handles MyBase.Load
            Try
            If Not (Request.QueryString("ItemId")
                            Is Nothing) Then
            ItemId=Int32.
               Parse(Request.QueryString("ItemId"))
            End If
            If Page.IsPostBack = False Then
               cmdDelete.Attributes.Add("onClick",
                   "javascript:return confirm('" &
            Localization.GetString("DeleteItem") & "');")
            If Not Common.Utilities.Null.IsNull(ItemId)
              Then
                Dim objCSEs As New CSEController
                Dim objCSE As CSEInfo =
                 objCSEs.GetCSE(ModuleId, ItemId)
               If Not objCSE Is Nothing Then
                   ' leggo i valori
              txtContent.Text = objCSE.Content
              tbTitle.Text = objCSE.Title
             tbKeywords.Text = objCSE.Keywords
            ctlAudit.CreatedDate =
                          objCSE.CreatedDate.ToString
            Else
               Response.Redirect(NavigateURL(), True)
            End If
            Else
                   cmdDelete.Visible = False
                   ctlAudit.Visible = False
            End If
     End If
   Catch exc As Exception
            ProcessModuleLoadException(Me, exc)
   End Try
End Sub
Private Sub cmdCancel_Click(ByVal sender As Object,
ByVal e As EventArgs) Handles cmdCancel.Click
Try
             Response.Redirect(NavigateURL(), True)
            [...]
End Sub
Private Sub cmdDelete_Click(ByVal sender As Object,
```

DISTRIBUIAMO IL NOSTRO MODULO!

Se creare il modulo è stato abbastanza semplice, la distribuzione ed il deploy saranno un gioco da ragazzi.

Occorre come prima cosa modificare alcuni file:

- 01.00.00.SqlDataProvider: contiene le istruzioni per la creazione delle tabelle e stored procedure su SQL Server. Dovremo copiarci dentro il codice che abbiamo utilizzato in fase di realizzazione del modulo. Sono in pratica gli script per la creazione delle tabelle e delle stored procedure. Nella crezione del file ricordiamoci di sostituire [dbo].[NomeDatabse] con {databaseOwner}{objectQualifier}
- CSE.SqlDataProvider: contiene le informazioni per la registrazione del modulo sul database
- CSE.dnn: è un file XML che prende il nome di Distribution Definition e contiene informazioni sull'installazione del modulo. In questo file dovremo indicare se verranno distribuiti anche i sorgenti o se si forniranno gli assembly (nell'esempio lasceremo i file sorgente).
- Uninstall.SqlDataProvider: contiene gli script per disinstallare il modulo

Al solito trovate tutti i file in questione nel codice allegato. Fatto questo, creiamo sul nostro disco rigido una cartella che chiameremo CSE. Copiamo dentro tutti i file presenti in DesktopModules\CSE ed in App_Code\CSE (se abbiamo deciso di non rilasciare i sorgenti, al posto dei file .cs dovremo copiare gli assembly) Creiamo infine un file CSE.zip da questa cartella. Il package a questo punto è pronto.
Per installare il modulo occorre:
- Accedere al portale
- Loggarsi come host
- Selezionare dal menu "Host->Module Definitions"
- Cliccare su "Upload New Module"
- Selezionare tramite "Sfoglia" il file zip e cliccare su "Add"
- Cliccare su "Upload New File"

CONCLUSIONI

DotNetNuke è sicuramente uno strumento molto potente già nella sua versione base. La possibilità di installare dei moduli aggiuntivi realizzati da terzi o di svilupparne dei propri, è un notevole punto a suo favore.
Lo sviluppo dei moduli è, come abbiamo visto, abbastanza indolore dal punto di vista dello "sforzo" di programmazione, anche grazie all'architettura multilivello di DotNetNuke.
Un prodotto da provare sicuramente, anche solo per il gusto di farlo!

Carmelo Scuderi

DEPLOYMENT DI APPLICAZIONI OFFICE

ABBIAMO CREATO UN SOFTWARE CHE ESTENDE WORD O EXCEL. NON CI RIMANE CHE CAPIRE COME POTERLO DISTRIBUIRE. UTILIZZEREMO QUALCHE TECNICA CHE CI CONSENTIRÀ DI MANTENERE LA DISTRIBUZIONE AGGIORNATA ALLE ULTIME VERSIONI...

Nel numero scorso abbiamo visto quanto sia semplice lo sviluppo di un'applicazione in grado di estendere le funzionalità dei normali documenti della suite Office utilizzando Visual Studio Tools for Office (VSTO), l'ambiente di sviluppo che consente di trattare i documenti Excel o Word come container di applicazioni .NET. Purtroppo la fase di distribuzione di questo genere di applicazioni non è altrettanto semplice. Microsoft mette comunque a disposizione degli sviluppatori diverse alternative, ognuna delle quali mostra i propri pregi ed i propri difetti. È infatti possibile distribuire le applicazioni VSTO attraverso la creazione di classici setup con Windows Installer oppure sfruttando la più moderna tecnica offerta da ClickOnce, la tecnologia che consente la rapida distribuzione di applicazioni con il minimo impatto sul sistema operativo client. Nella versione attuale, quella integrata in Visual Studio 2005, la distribuzione di questo tipo di applicazioni tuttavia non è ancora pienamente supportata. Proveremo in questo articolo ad analizzare le difficoltà di questo modello e a trovare le possibili alternative.

DEPLOYMENT MODELS

Indipendentemente dalla modalità scelta, il deploy di un'applicazione VSTO necessita essenzialmente di quattro files:

- Il documento Word o Excel che stiamo distribuendo;
- L'assembly associato al documento e che contiene il codice compilato
- Il deployment manifest che riporta informazioni riguardanti la versione attualmente distribuita;
- L'application manifest che racchiude le informazioni sulle singole versioni degli assembly distribuiti;

Oltre ai file sopra indicati, dobbiamo tenere presente quelli che sono i prerequisiti necessari e indispensabili per il corretto funzionamento sul computer client. La distribuzione di VSTO via ClickOnce, infatti, assume che sul client siano già installate le seguenti applicazioni:

- Il .NET Framework 2.0
- Il pacchetto Microsoft Office 2003 Service Pack 2 oppure una versione standalone di Word o Excel
- Il runtime di Visual Studio Tools for Office runtime ed eventualmente il Language Pack
- Il pacchetto Microsoft Office 2003 Primary Interoperable Assembly

Oltre alle caratteristiche evidenziate è necessario che sia il documento Word o Excel sia l'assembly ad esso collegato riceva il set di permessi *FullTrust* dalla Code Access Security (CAS).
Il modello di deploy adottato può prevedere tre diverse modalità, variando a seconda del path di esecuzione del documento o dell'assembly collegato:

- Local/Local: consente la distribuzione di copie del documento e degli assembly collegati su ogni singolo computer. Questa modalità consente di lavorare offline ed è l'ideale in caso di indisponibilità della rete;
- Local/Network: questo modello consente di avere la copia locale del documento office, ma di avere l'assembly collegato su una condivisione di rete. Anche in questa modalità è possibile avere la totale disponibilità sia del documento sia dell'assembly se si utilizza un protocollo http/https, dato che viene posto in cache e ripreso al successivo caricamento;
- Network/Network: in questa modalità sia il documento office sia l'assembly ad esso collegato sono posizionati in una condivisione di rete che sia essa una cartella condivisa, un percorso ftp o un percorso http;

Sostanzialmente il deploy avviene copiando manualmente i files nei diversi path. È possibile però automatizzare il processo di pubblicazione

delle applicazioni VSTO sfruttando il Publish Wizard che ci dà accesso alle funzionalità di ClickOnce.
Il *Publish Wizard*, infatti, crea per noi l'application ed il deployment manifest e copia tutti i files nel path da noi selezionato. Ma seguiamo per gradi le fasi di creazione e di pubblicazione di una applicazione VSTO.

CREIAMO UNA NUOVA APPLICAZIONE VSTO

Il setup di VSTO installa alcuni nuovi template disponibili in Visual Studio per la creazione delle diverse tipologie di soluzioni. Per creare un nuovo progetto VSTO apriamo Visual Studio 2005, selezioniamo *ExcelWorkbook* ed indichiamo nel nome del progetto *ExcelApplication* e clicchiamo su OK.

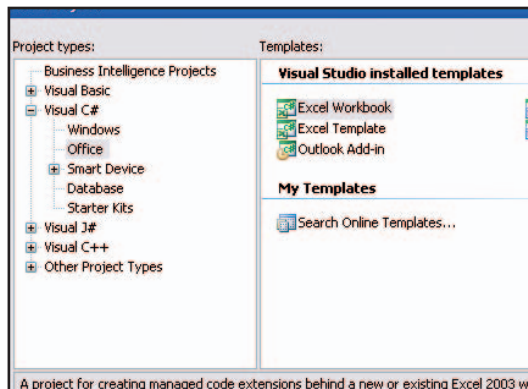



Fig. 1: Creazione di un nuovo progetto VSTO

A questo punto Visual Studio genera il progetto sulla base del template selezionato, creando un Excel Workbook con, di default, tre fogli Excel. Apriamo il primo foglio Excel e inseriamo un controllo *NamedRange*, che ci permette di definire un range di celle sulle quali poi andremo ad operare dal nostro codice. Inseriamo ora un controllo di tipo button per ottenere un foglio Excel simile a quello visualizzato in figura 2.

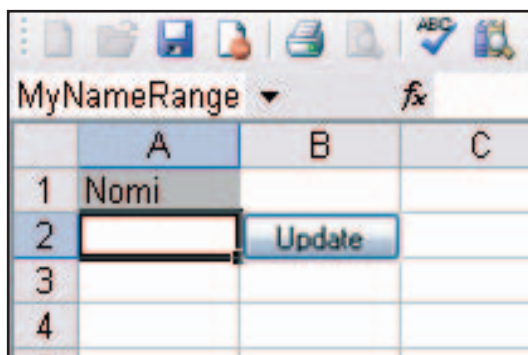



Fig. 2: un semplice documento con un pulsante

A scopo puramente dimostrativo, il documento Excel mostrerà, al clic sul mouse, un nome differente tra quelli precedentemente inseriti effettuando un binding verso un oggetto di tipo dataset. Per realizzare l'applicazione dichiariamo le seguenti variabili di classe:

```
private string[] Redattori = { "Fabio",
   "Domenico", "Luca", "Anna", "Rossana" };
private DataSet ds;
```

inseriamo poi nell'evento *Foglio1_Startup* il seguente codice:

```
ds = new DataSet();
DataTable table =
            ds.Tables.Add("Redattori");
DataColumn column1 = new
      DataColumn("Nomi", typeof(string));
table.Columns.Add(column1);
// Riempie le righe della tabella
// Con i valori presi dall'array redattori
DataRow row;
for (int i = 0; i < Redattori.Length; i++)
{
    row = table.NewRow();
    row["Nomi"] = Redattori[i];
    table.Rows.Add(row);
}
// Crea un "bind" fra la property
//  Value2 del
// NamedRanged e il contenuto
//della tabella nomi
// Dell'array redattori
Binding binding1 = new
Binding("Value2",  ds, "Redattori.Nomi",
                         false);
MyNameRange.DataBindings.Add(binding1);
```

Nell'evento clic del *button* inseriamo, invece, il seguente codice:

```
if (MyNameRange.BindingContext != null)
{
    BindingManagerBase bindingManager1 =
               MyNameRange.BindingContext
                       [ds, "Redattori"];
        // Mostra il contenuto del
       // prossimo record

        if (bindingManager1.Position <
              bindingManager1.Count - 1)
        {
            bindingManager1.Position++;
        }

        // Mostra il primo record
        else
```

```
            {
                bindingManager1.Position = 0;
            }
        }
```

Il risultato rappresenta una variazione a quanto riportato in un precedente articolo su VSTO.

DISTRIBUIAMO IL DOCUMENTO OFFICE

L'architettura supportata da ClickOnce prevede l'esecuzione dell'applicazione in una modalità sicura, soggetta alla CAS (Code Access Security) del .NET Framework e che può quindi essere eseguita solo richiedendo il set minimo di permessi. Di contro, però, la versione attuale di VSTO richiede obbligatoriamente l'attribuzione del set di permessi *FullTrust* sia per il documento office sia per l'assembly .NET. Per risolvere il problema dobbiamo provvedere a rendere sicuro, affidabile e certo l'assembly da distribuire, e poi dobbiamo rendere "trusted" il percorso di esecuzione dell'assembly stesso. ClickOnce consente la distribuzione delle applicazioni utilizzando diverse modalità come la distribuzione via sito web utilizzando il protocollo http, l'utilizzo di un percorso ftp oppure di una semplice condivisione di rete. Nel nostro scenario ipotizziamo la distribuzione in una Intranet utilizzando una cartella condivisa. Per rendere sicuro il nostro assembly dobbiamo, per prima cosa, firmarlo con uno *strong name*: da Visual Studio 2005 clicchiamo con il tasto destro sul nome del progetto *ExcelApplication* e selezioniamo *Properties*. Nella finestra delle proprietà abbiamo a disposizione diverse schede e sulla sinistra clicchiamo sulla scheda *Signing* e impostiamo il check *Sign the assembly* per consentirci di firmare digitalmente l'assembly. Nella combo box scegliamo *<new>* per creare un nuovo file di chiavi, impostiamo il nome *vstokey.snk* senza indicare alcuna password e clicchiamo su ok. Se tutto è andato a buon fine il risultato sarà quello visualizzato in figura 3.

Al termine della procedura di configurazione, Visual Studio 2005 crea un nuovo file vstokey.snk che utilizzerà in fase di compilazione per firmare l'assembly. A questo punto siamo pronti per eseguire la compilazione e pubblicare il risultato nel percorso condivisa. Avviamo la compilazione e al termine procediamo con la pubblicazione. Facciamo clic con il tasto destro del mouse sul nome del progetto *ExcelApplication* e clicchiamo su *Publish* o *Pubblica* per chi ha la versione italiana di Visual Studio 2005. A questo punto parte il wizard che ci chiede quale modalità di pubblicazione vogliamo utilizzare. Scegliamo un percorso di rete indicando ***\\server\cartella***, clicchiamo su next e poi su *finish* per eseguire la pubblicazione. Contestualmente viene rieseguita la compilazione, vengono creati il deployment manifest e l'application manifest ed infine viene tutto copiato nel path indicato. Ora dovremmo essere in grado di eseguire l'applicazione semplicemente avviandola dal path indicato. Prima, però, come già indicato dobbiamo necessariamente assegnare, su tutti i computer, i giusti permessi di esecuzione alla cartella condivisa.

E LA SICUREZZA?

Come sopra descritto, un'applicazione VSTO per essere eseguita correttamente necessita di acquisire un set di permessi di tipo FullTrust. Al fine di assegnare i giusti permessi dobbiamo creare un gruppo di codice (*Code Group*) in grado di garantire il set *FullTrust*, appunto, agli assembly in esecuzione nel path che successivamente andremo ad indicare. Per far questo dobbiamo eseguire alcuni semplici passi:

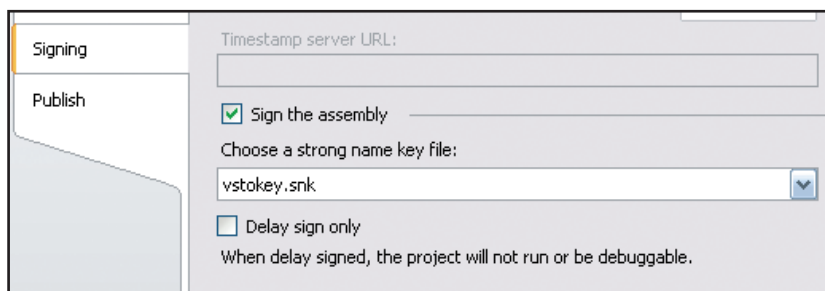



Fig. 3: *La finestra delle proprietà del progetto.*

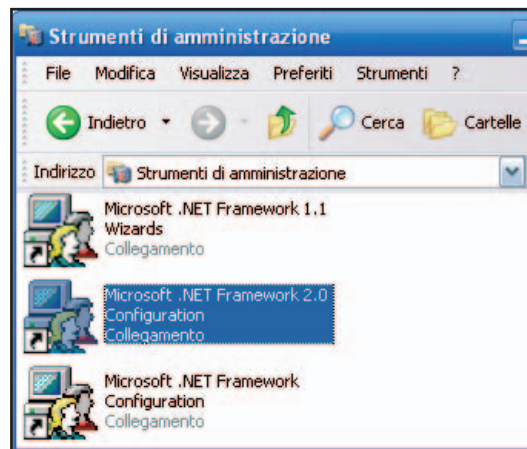



Fig. 4: *Il .NET Framework Configuration Tool.*

1 Apriamo il *.NET Framework Configuration Tool* dal Pannello di Controllo/Strumenti di Amministrazione.

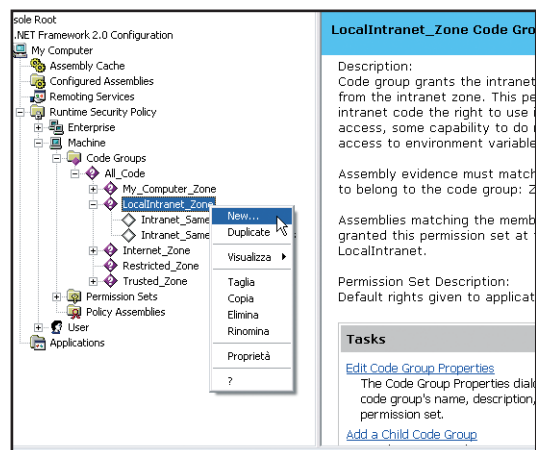



Fig. 5: **La creazione di un nuovo code group**

2 Espandiamo il nodo *My Computer/Runtime Security Policy/Machine/Code Groups /All_Code/LocalIntranet_Zone*. Qui clicchiamo con il tasto destro del mouse e selezioniamo New per avviare il wizard di creazione del nuovo gruppo di codice.

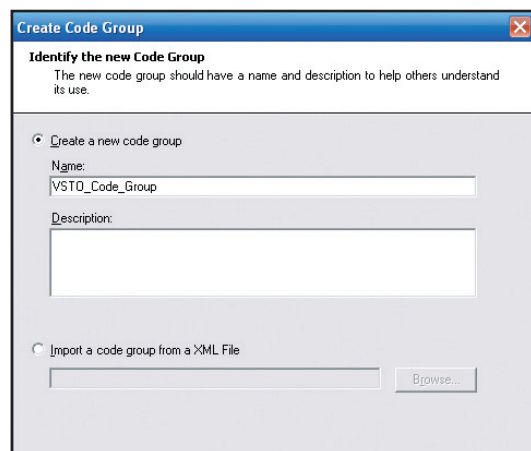



Fig. 6: **Selezioniamo il nome del gruppo di codice.**

3 Nella prima scheda del wizard indichiamo il nome del gruppo di codice, ad esempio *VSTO_Code_Group*, e clicchiamo su *Next*.

4 Successivamente indichiamo la condizione che l'assembly deve rispettare per ottenere i permessi assegnato a questo specifico gruppo di codice. Qui selezioniamo come tipo di condizione l'appartenenza ad un URL, che indichiamo nel secondo campo di input come ***\\server\cartella***, ad esempio ***\\fabioc\publish***, e clicchiamo su Next.
L'asterisco indicato e visibile in figura 7 serve per consentire i permessi a tutti i files in esecuzione nella specifica cartella e nelle sue sottocartelle

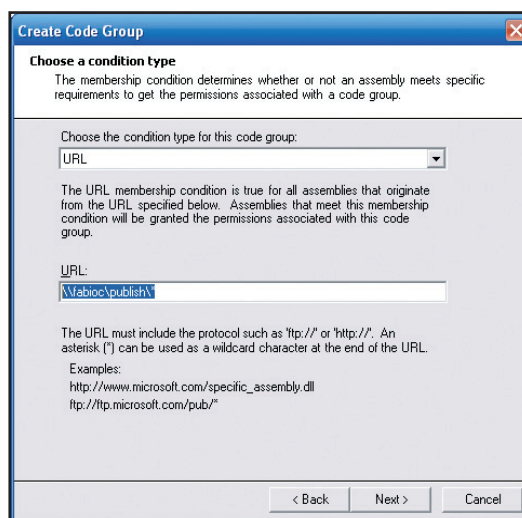



Fig. 7: **Impostiamo il tipo di condizione da rispettare**

5 Infine impostiamo il *Permission Set* che il gruppo di codice garantisce agli assembly che rispettano la condizione precedentemente impostata. Nel nostro caso assegniamo *FullTrust*, clicchiamo su *Next* e poi su *Finish*.
La procedura indicata consente di creare il giusto gruppo di codice con il set di permessi di tipo *FullTrust* che come abbiamo già detto è necessario per l'esecuzione di applicazioni VSTO. È importante sottolineare che bisogna essere amministratori della macchina per poter eseguire le operazioni sopra descritte.

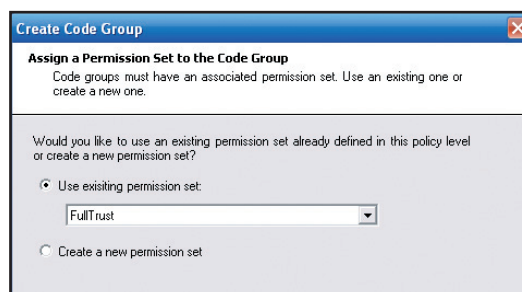



Fig. 8: **Indichiamo FullTrust come permission set.**

LA CODE ACCESS SECURITY

La CAS è presente fin dalla prima versione del Microsoft .NET Framework e consente di prevenire l'esecuzione di codice non attendibile o che richiede elevati permessi per essere eseguito. Il runtime verifica, al caricamento di un assembly, la sua attendibilità sulla base della cosiddetta evidence, calcolata utilizzando parametri come la firma dell'assembly via strong-name o certificato digitale, la sua zona di esecuzione, ecc..., utilizzandola per identificare il gruppo di codice di appartenenza e i relativi permessi di esecuzione. I gruppi di codice (code group) vengono generalmente determinati dagli amministratori di sistema. Se i permessi richiesti non coincidono con il gruppo di codice di riferimento, il runtime genera una eccezione di sicurezza.

L'ESECUZIONE SU UN COMPUTER DIVERSO

La procedura di pubblicazione ha generato una particolare struttura di files e cartelle secondo le regole stabilite dalla tecnologia ClickOnce. In figura 9 è visualizzata una struttura di cartelle che mostra nella root il documento excel ed il deploy manifest, mentre nelle varie sottocartelle, contenente l'application manifest e gli assembly dipendenti, una suddivisione per versioni. È il deployment manifest che indica qual è la versione correntemente in uso e lo fa in due modi:

1. Indicando la evidence del deployment manifest:

```
<assemblyIdentity
    name="ExcelApplication.application"
    version="1.0.0.7"
    publicKeyToken="0000000000000000"
    language="neutral" processorArchitecture="msil"
    xmlns="urn:schemas-microsoft-com:asm.v1" />
```

2. Indicando la evidence degli assembly dipendenti:

```
<dependency>
<dependentAssembly dependencyType="install"
codebase="ExcelApplication_1.0.0.7\ExcelApplication.
    dll.manifest" size="1197">
<assemblyIdentity name="ExcelApplication.dll"
    version="1.0.0.7" />
```

Se proviamo ad eseguire il documento Excel contenuto nella root su un pc differente da quello di sviluppo, funzionerà tutto correttamente. Possiamo anche prelevare il file e copiarlo su una directory differente sul pc locale e tutto continuerà a funzionare regolarmente. Come avviene? Semplicemente il documento Excel mantiene un riferimento alla cartella condivisa, dalla quale recupera ed esegue gli assembly dipendenti. Questo meccanismo consente di avere la certezza di eseguire sempre la versione ultima del documento.

INSTALLAZIONE DEI PREREQUISITI

Prima di poter eseguire qualsiasi applicazione VSTO, è necessario che sul computer dell'utente siano presenti alcuni prerequisiti procedendo come di seguito:

1. **Installare il .NET Framework**
2. **Installare il runtime di Visual Studio Tools for Office, prelevabile da questo indirizzo:** http://go.microsoft.com/fwlink/?linkid=49612&clcid=0x409
3. **Opzionalmente è possibile installare il Visual Studio Tools for Office Language Pack che consente la visualizzazione dei messaggi nella stessa lingua del sistema operativo. Il VSTO Language Pack è disponibile nell'area download** http://www.microsoft.com/downloads
4. **Installare i Primari Interoperable Assembly disponibili nel setup di Office 2003**

Per il punto 4 in particolare è importante selezionare la voce .NET Programmability Support nel wizard di installazione di Office 2003 Professional, così da consentire l'installazione dei Primary Interoperable Assembly, il set di assembly che contengono la definizione dei tipi che sono stati definititi negli oggetti COM.

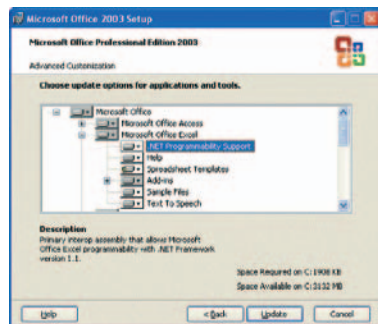



Fig. 10: Il wizard di installazione di Office 2003

NUOVA RELEASE

Dopo aver analizzato le diverse problematiche, proviamo ora ad eseguire un update dell'applicazione pubblicandone una nuova versione. Riprendiamo il progetto *Excel Application* e proviamo ad eseguire una qualsiasi modifica, ad esempio aggiungendo un qualsiasi nuovo elemento nell'array dei nomi dei redattori. Al termine delle modifiche rieseguiamo il *Publish Wizard* confermando tutte le precedenti impostazioni. Proviamo ora ad eseguire il documento Excel aggiornato e a cliccare sul pulsante *Update* per scorrere i risultati e fino a visualizzare il nuovo redattore inserito. Ovviamente se modifichiamo la struttura del documento andremo a sovrascrivere il documento precedentemente pubblicato. Per evitare questo problema potrebbe essere una buona soluzione utilizzare il foglio Excel per ottenere informazioni da fonti di dati come, ad esempio, database Sql Server 2005.

CONCLUSIONI

Nel presente articolo abbiamo visto come creare applicazioni utilizzando Visual Studio Tools for Office, ma soprattutto come distribuire le applicazioni create verso gli utenti che ne faranno poi uso. Abbiamo poi analizzato le varie problematiche legate alla distribuzione di documenti Office e come è possibile risolverle. Come già accennato, purtroppo l'integrazione tra ClickOnce e VSTO non è ancora perfetta, ma lo sarà senza dubbio nelle prossime versioni.

Fabio Cozzolino

GESTISCI AL MEGLIO LE TUE RISORSE

AVETE A DISPOSIZIONE 5 SCIALUPPE DI SALVATAGGIO E 50 PERSONE DA SALVARE. OGNI SCIALUPPA PUÒ PORTARE AL MASSIMO 400KG. POICHÈ IL PESO DELLE PERSONE È MOLTO VARIABILE, COME POSSO DISTRIBUIRLE IN MODO OTTIMALE?

Spesso capita di avere a che fare con problemi che appaiono inizialmente molto semplici, ma che in realtà celano una complessità notevole. Si pensi, ad esempio, di dover caricare 50 persone, 50 per un peso complessivo di 4000 kg, su un certo numero di scialuppe, che possiamo supporre essere 11 con una portata di 400 kg ciascuna. Se ipotizzassimo che le persone pesano tutte allo stesso modo, e quindi 80 kg ciascuna, potremmo pensare di mettere 5 persone su ogni scialuppa, lasciandone addirittura una vuota. Ma la supposizione non ha alcun fondamento e pertanto, la soluzione pensata non ha alcuna validità. Basta pensare che, se il nostro campione fosse composto da 49 persone di 81 kg più un bambino di 31 kg, in ogni scialuppa potremmo caricare al più 4 persone, ed in una scialuppa 4 persone più il bambino, per un totale di sole 45 persone; in questo caso il nostro problema non ammetterebbe alcuna soluzione ovvero la soluzione sarebbe quella di "sacrificare" 5 persone.

UNA COMPLESSITÀ DECISAMENTE ELEVATA

Il problema presentato è molto simile ad un problema di knapsack multiplo, in cui si hanno a disposizione *m* contenitori di capacità *wi* ed *n* oggetti con un peso *pj*; è importante precisare che il problema del knapsack è un problema NP-Completo, ovvero tra i più difficili da risolvere, e, nella sua versione decisionale rientra tra i 21 problemi NP-Completi di Karp. Volendo, tuttavia, distribuire comunque tutti gli oggetti nei vari contenitori, oltre a dover imporre che la capacità complessiva dei contenitori sia maggiore o uguale al peso degli oggetti da allocare, e quindi funzione di essi, possiamo ammettere la presenza nella soluzione di gap e overlap, cioè di non riempire completamente un contenitore oppure di allocarvi oggetti in misura superiore alla sua capacità. Queste ipotesi di fatto trasformano il nostro problema in un problema di partizionamento: un esempio può essere la distribuzione dei clienti di una società su canali commerciali in modo proporzionale alla capacità degli stessi canali; nel nostro caso, prenderemo come esempio il problema di una società che ha la necessità di incassare tramite disposizioni bancarie, importi in addebito ai propri clienti, suddividendo la totalità degli importi su più conti secondo un qualche criterio; la suddivisione potrebbe essere fatta sulla base dei castelletti salvo buon fine definiti sulle varie banche; tuttavia, esistendo l'esigenza di incassare tutti gli importi dovuti dai clienti, si può pensare di adottare un criterio di suddivisione percentuale sull'importo complessivo.

$$\begin{cases} \max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} & \\ \sum_{j=1}^{n} p_j x_{ij} <= w_i & i=1..m \\ \sum_{i=1}^{m} x_{ij} <= 1 & j=1..n \\ x_{ij} = \{0, 1\} & i=1..m\ ,\ j=1..n \end{cases} \qquad \begin{cases} \sum_{j=1}^{n} p_j x_{ij} \approx w_i & i=1..m \quad w_i=f(p_1,..,p_n) \\ \sum_{i=1}^{m} x_{ij} = 1 & j=1..n \\ x_{ij} = \{0, 1\} & i=1..m\ ,\ j=1..n \end{cases}$$

Fig. 1: La formulazione matematica del knapsack multiplo e la trasformazione per modellare il nostro problema; la funzione obiettivo si esaurisce nella definizione delle capacità e nella modifica del secondo vincolo, l'ammissibilità della soluzione è garantita dall'elasticità informale del primo vincolo.

IMPOSTAZIONE

La difficoltà intrinseca in un problema di questo tipo è dovuta ad una serie di particolarità, quali la granularità; si supponga infatti di non poter suddividere gli importi di ciascun cliente, ad esempio per ridurri i costi delle operazioni bancarie; il dominio del problema è dunque un dominio discreto e pertanto non sempre esiste la soluzione esatta. Dovendoci accontentare, in molti casi, di una soluzione sub-ottima, e vista la complessità del problema, è senza dubbio opportuno affidarsi ad un algoritmo di tecnica "golosa" (greedy). Supponiamo inoltre di dover elaborare un numero sufficientemente grande di clienti e di voler generare sulla base dati le disposizioni ed i supporti logici -da cui estrarre in un secondo momento i file RID da inviare alle banche lavorando i dati direttamente sul DB, senza utilizzare una

applicazione di appoggio, ad esempio per evitare lo scarico dei dati iniziali e l'upload dei dati post-elaborazione. Come ambiente di sviluppo verrà utilizzato SQL Server 2000, ed in particolare i cursori con le estensioni T-SQL.
Per testare la procedura, è possibile utilizzare i dati di esempio presenti nell'allegato; si tratta di un set di poco più di 16.000 transazioni effettuate da un campione di 1.000 clienti su un parco di 13 prodotti; l'importo delle singole transazioni è stato per comodità considerato variabile e non legato al prodotto, con una serie di importi che variano da € 2,02 ad € 1.158,99; avremo quindi 1.000 disposizioni che variano da un totale di € 1.360,00 ad un totale di € 17.343,84 da allocare su 4 banche secondo uno schema percentuale 50-30-15-5.

LA TECNICA GREEDY

Ritornando momentaneamente al problema del knapsack, che è del tutto simile al nostro per quando riguarda la definizione dei vincoli, e differisce solo per la funzione obiettivo, che nel nostro caso viene "adattata" al problema per far spazio al vincolo di completa assegnazione delle risorse, gli algoritmi greedy affrontati in letteratura si basano fortemente su tecniche di ordinamento degli oggetti.
Come vedremo, anche la nostra tecnica si basa sull'ordinamento, sebbene non fornisce alcuna garanzia sulla "qualità" del risultato. Si pensi banalmente al caso "degenere" di un solo cliente, e quindi di una sola disposizione: in questo caso gli importi saranno inviati ad una sola banca che riceverà il 100% del totale, mentre le altre banche riceveranno lo 0%. Occorre comunque precisare che quando la cardinalità del problema in termine di clienti, e quindi di disposizioni è sufficientemente grande e gli importi totali delle disposizioni hanno una distribuzione casuale, i risultati della tecnica si rivelano piuttosto validi. Innanzitutto, come dovrebbe essere già chiaro a questo punto, gli oggetti da allocare non sono le singole transazioni, ma le disposizioni, cioè il totale delle transazioni di ciascun cliente. Pertanto, un primo passo, che non è proprio dell'algoritmo, dovrà prevedere la creazione dei dati aggregati per ciascun cliente. Successivamente, possiamo impostare l'algoritmo nel seguente modo: possiamo ordinare le disposizioni dalla più grande alla più piccola in termine di importi- e anche le banche, da quella che dovrà ricevere una quantità maggiore di importi a quella che dovrà riceverne meno.
Quindi, partendo dalla disposizione più onerosa, si tenta di allocarla sulle banche, a partire da quella più ricettiva. Per meglio comprendere come funziona l'algoritmo, eseguiamo i passi su una istanza più piccola rispetto a quella allegata e gestibile manualmente, come quella di figura 1; in particolare consideriamo 2 banche (70%, 30%) e 9 disposizioni da allocare, di importo compreso da €10,00 ad € 90,00 con un passo di € 10,00. Come si può subito osservare, non esiste la possibilità di ottenere una soluzione esatta, essendo gli importi delle disposizioni multipli di € 10,00 a fronte di una ripartizione richiesta in € 315,00 sulla banca A e € 135,00 sulla banca B. Ordinando le entità secondo i criteri descritti, dovremo considerare prima la banca A, successivamente la banca B; per le disposizioni invece partiremo dalla D9 fino ad arrivare alla D1. Pertanto, seguendo la prima parte dell'algoritmo risulterà semplice allocare le disposizioni D9-D3,

Banche	A	B
	70%	30%

Disposizioni	D1	D2	D3	D4	D5	D6	D7	D8	D9
	€ 10,00	€ 20,00	€ 30,00	€ 40,00	€ 50,00	€ 60,00	€ 70,00	€ 80,00	€ 90,00

Fig. 2: ***Esempio di istanza***

come da figura:
Per le disposizioni D9-D6 infatti, troveremo sempre posto nella banca A, mentre le disposizioni D5-D3, essendo superiori alla capacità residua della banca A, verrano destinate alla banca B. Al momento di selezionare la banca per D2, vediamo però che

Banca	A	B
Percentuale richiesta	70%	30%
Importo richiesto	€ 315,00	€ 135,00
Percentuale attuale	66,67%	26,67%
Importo attuale	€ 300,00	€ 120,00

Disposizioni	D1	D2	D3	D4	D5	D6	D7	D8	D9
Importo	€ 10,00	€ 20,00	€ 30,00	€ 40,00	€ 50,00	€ 60,00	€ 70,00	€ 80,00	€ 90,00
Banca destinataria	?	?	B	B	B	A	A	A	A

Fig. 3: ***Soluzione parziale***

ovunque questa venga posta, si avrebbe un superamento degli importi richiesti, come si può vedere dalla figura che illustra due delle quattro possibili combinazioni, avendo escluso quelle che vedono entrambe le disposizioni assegnate alla stessa banca: La scelta della banca destinataria può quindi essere effettuata o su base casuale, o, nel tentativo di ridurre gli errori, con un criterio euristico. Nel nostro caso decidiamo di effettuare la scelta che minimizza l'errore relativo (in valore assoluto); per la disposizione D2, se questa venisse destinata alla banca A, porterebbe ad un totale di € 320,00 contro gli € 315,00 deisderati, con un errore relativo di (320-315)/315 = 1,58%. Mentre, se venisse destinata alla banca B, porterebbe ad un totale di € 140,00 contro gli € 135,00 richiesti, con un errore relativo di (140-135)/135 = 3,70%; la disposizione D2 viene pertanto destinata alla banca A.
Iterando il procedimento sulla disposizione D1, avremmo sulla banca A un errore relativo di (330-315)/315 = 4,76%, mentre sulla banca B un errore relativo di (135-130/135) = 3,70%; la soluzione trovata pertanto è la soluzione 1 di figura 4.

Soluzione 1

Banca	A	B
Percentuale richiesta	70%	30%
Importo richiesto	€ 315,00	€ 135,00
Percentuale attuale	71,11%	28,89%
Importo attuale	€ 320,00	€ 130,00

Disposizioni	D1	D2	D3	D4	D5	D6	D7	D8	D9
Importo	€ 10,00	€ 20,00	€ 30,00	€ 40,00	€ 50,00	€ 60,00	€ 70,00	€ 80,00	€ 90,00
Banca destinataria	B	A	B	B	B	A	A	A	A

Soluzione 2

Banca	A	B
Percentuale richiesta	70%	30%
Importo richiesto	€ 315,00	€ 135,00
Percentuale attuale	68,89%	31,11%
Importo attuale	€ 310,00	€ 140,00

Disposizioni	D1	D2	D3	D4	D5	D6	D7	D8	D9
Importo	€ 10,00	€ 20,00	€ 30,00	€ 40,00	€ 50,00	€ 60,00	€ 70,00	€ 80,00	€ 90,00
Banca destinataria	A	B	B	B	B	A	A	A	A

***Fig. 4:** Le due possibili soluzioni*

LA STRUTTURA DATI

Per costruire il nostro algoritmo T-SQL, faremo ricorso a strutture dati piuttosto semplificate; utilizzeremo la tabella CMDACCTransaction che conterrà le transazioni di ciascun cliente e la tabella CMDDATBankAccount che conterrà le anagrafiche delle banche destinatarie:

```
CREATE TABLE [dbo].[CMDACCTransaction] (
[Id] [int] IDENTITY (1, 1) NOT NULL ,
[ClientCode] [char] (7) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[ProductCode] [char] (5) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[OperationDate] [datetime] NOT NULL ,
[Amount] [money] NOT NULL ,
[CauseDescription] [varchar] (50) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[DispositionId] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[CMDDATBankAccount] (
```

***Fig. 5:** Diagramma entità-relazioni*

```
[Id] [int] IDENTITY (1, 1) NOT NULL ,
[ABICode] [char] (5) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[CABCode] [char] (5) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[AccountNumber] [char] (12) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[CIN] [char] (1) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[IBAN] [varchar] (50) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[Percentage] [float] NOT NULL ,
[Enabled] [bit] NOT NULL
) ON [PRIMARY]
GO
```

Per maggior efficienza possiamo supporre di avere, sulla tabella CMDACCTransaction, un indice clusterizzato sul campo ClientCode. Inoltre utilizzeremo le tabelle CMDRIDD isposition e CMDRIDSupport che conterranno rispettivamente le disposizioni di pagamento ed i supporti logici, intesi come contenitori di disposizioni:

```
CREATE TABLE [dbo].[CMDRIDDisposition] (
    [Id] [int] IDENTITY (1, 1) NOT NULL ,
    [SupportId] [int] NOT NULL ,
    [Progressive] [int] NOT NULL ,
    [ClientCode] [char] (7) COLLATE
        SQL_Latin1_General_CP1_CI_AS NOT NULL ,
    [Amount] [money] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[CMDRIDSupport] (
    [Id] [int] IDENTITY (1, 1) NOT NULL ,
    [BankAccountId] [int] NOT NULL ,
    [CreationDate] [datetime] NOT NULL ,
    [DispositionsNumber] [int] NOT NULL ,
    [Amount] [money] NOT NULL ,
    [Completed] [bit] NOT NULL
) ON [PRIMARY]
GO
```

Nel diagramma che segue sono mostrate le relazioni tra le varie tabelle; per completezza è stata aggiunta la tabella CMDDATClient che raccoglie le anagrafiche dei clienti.

USO DEI CURSORI

Come si può osservare, la natura dell'algoritmo descritto è iterativa, pertanto è necessario ricorrere a tecniche di iterazione sui dati, nel nostro caso recordset; in tal senso ci vengono incontro i cursori in T-SQL.
Utilizzando la sintassi:

```
DECLARE cursor_name CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR select_statement
```

è possibile definire un cursore, cioè una struttura dati su cui sono disponibili operazioni di iterazione, su una query; a seconda della definizione del cursore, sarà possibile effettuare su di esso una o più operazioni di fetch (next, prior, last, first, relative, absolute), ovvero di scrolling del cursore sul recordset. L'algoritmo prevede una iterazione esterna sulle disposizioni che ne prevede l'immediata assegnazione ad un determinato supporto; su questo recordset pertanto non sono richieste né la dinamicità dei dati, né operazioni di fetch oltre l'avanzamento. Pertanto possiamo definire il cursore FORWARD_ONLY STATIC, e impostare lo scheletro della procedura come segue:

```
DECLARE @ClientCode                    AS CHAR(7)
DECLARE @DispositionAmount  AS MONEY
...
DECLARE DispositionCursor CURSOR LOCAL
                          FORWARD_ONLY STATIC FOR
SELECT   T.ClientCode, SUM(ROUND(T.Amount, 2))
FROM            dbo.CMDACCTransaction T
WHERE    T.Amount > 0
AND             T.DispositionId IS NULL
GROUP           BY T.ClientCode
ORDER           BY SUM(ROUND(T.Amount, 2))
                                          DESC
OPEN SupportCursor
OPEN DispositionCursor
FETCH NEXT FROM DispositionCursor INTO
                @ClientCode, @DispositionAmount
WHILE (@@FETCH_STATUS = 0)
BEGIN
        --ricerca supporto di destinazione
        --crea disposizione e aggiorna supporto
FETCH NEXT FROM DispositionCursor INTO
                @ClientCode, @DispositionAmount
END
```

Per la ricerca della banca di destinazione, ovvero del supporto, abbiamo invece la necessità di operazioni di fetch di avanzamento e di riposizionamento del cursore ad inizio recordset; inoltre, supponendo di aver inizialmente creato un supporto vuoto per ciascuna banca destinataria, ci interessa conoscere l'attuale stato di riempimento del supporto, e pertanto sarà necessario adoperare un cursore *SCROLL DYNAMIC,*:

```
DECLARE ...
...
DECLARE SupportCursor CURSOR LOCAL SCROLL
                                  DYNAMIC FOR
SELECT   RS.Id, BA.Percentage / 100, RS.Amount,
                  RS.DispositionsNumber FROM
                  dbo.CMDRIDSupport RS
                  JOIN dbo.CMDDATBankAccount
                  BA ON RS.BankAccountId = BA.Id
WHERE    RS.Completed = 0 ORDER BY BA.Percentage
                                          DESC
...
--Ricerca supporto destinazione
SET @BankAccountFound = 0
SET @RelativeErrorMax = -1
FETCH FIRST FROM SupportCursor INTO @SupportId,
           @SupportPercentage, @SupportAmount,
                        @DispositionsNumber
        WHILE (@@FETCH_STATUS = 0) AND
                        (@BankAccountFound = 0)
                        BEGIN
                        SET
      @SupportNewPercentage = (@SupportAmount +
             @DispositionAmount) / @TotalAmount
--Rientra nei parametri del supporto: conto trovato
             IF  @SupportNewPercentage
                 <= @SupportPercentage
                  BEGIN
                     SET  @SupportGreedyId =
                                @SupportId
                     SET  @DispositionProgressive
                       = @DispositionsNumber + 1
                     SET  @SupportNewAmount =
                          @SupportAmount +
                          @DispositionAmount
                     SET @BankAccountFound = 1
                END
                ELSE

                --Calcola errore e verifica se
                -- il supporto corrente è la
                -- migliore alternativa

                BEGIN SET
         @RelativeError = (@SupportNewPercentage -
         @SupportPercentage) / @SupportPercentage
IF (@RelativeErrorMax < 0) OR (@RelativeError <
                              @RelativeErrorMax)
      BEGIN
         SET @SupportGreedyId = @SupportId
         SET @DispositionProgressive =
                        @DispositionsNumber + 1
         SET @RelativeErrorMax = @RelativeError
         SET @SupportNewAmount =
             @SupportAmount + @DispositionAmount
      END
END
                FETCH NEXT FROM SupportCursor
             INTO @SupportId, @SupportPercentage,
```

```
                    @SupportAmount, @DispositionsNumber
END
--crea disposizione e aggiorna supporto
INSERT              INTO dbo.CMDRIDDisposition
              (ClientCode, SupportId, Progressive, Amount)
VALUES  (@ClientCode, @SupportGreedyId,
              @DispositionProgressive, @DispositionAmount)
SET     @DispositionId = @@IDENTITY
UPDATE  dbo.CMDRIDSupport
SET                 Amount = @SupportNewAmount,
                            DispositionsNumber =
                              @DispositionProgressive
WHERE   dbo.CMDRIDSupport.Id =
                                  @SupportGreedyId
UPDATE  dbo.CMDACCTransaction
SET                 DispositionId = @DispositionId
WHERE   ClientCode = @ClientCode
AND                 Amount > 0
AND                 DispositionId IS NULL
```

Completata l'assegnazione infine si prevedono le operazioni di chiusura dei cursori, di eliminazione dei supporti eventualmente vuoti –essendo stati creati preventivamente- e di flaggatura dei supporti come completati:

```
CLOSE     SupportCursor
DEALLOCATE SupportCursor

CLOSE     DispositionCursor
DEALLOCATE DispositionCursor
DELETE
FROM      dbo.CMDRIDSupport
WHERE     dbo.CMDRIDSupport.Completed = 0
AND       dbo.CMDRIDSupport.DispositionsNumber =
                                                0
UPDATE    dbo.CMDRIDSupport
SET       Completed = 1
WHERE     Completed = 0
```

UN ALGORITMO PIÙ EFFICIENTE

Attraverso l'utilizzo di cursori T-SQL si è realizzato un algoritmo strettamente legato al paradigma della programmazione imperativa. Tuttavia, l'ambiente utilizzato non nasce con queste finalità e le performance dell'algoritmo degradano al crescere delle disposizioni: è facile osservare che facendo aumentare il numero dei clienti, ad esempio a 10.000, si ottengono prestazioni che in alcuni ambiti possono essere ritenute non soddisfacenti.
È possibile ottimizzare l'algoritmo, a discapito della sua precisione, lasciando invariato l'approccio basato sull'ordinamento delle entità, ma sfruttando maggiormente le primitive più tipiche del linguaggio SQL. Prima di vedere i dettagli implementativi, illustriamo utilizzando il nostro esempio, l'algoritmo che verrà realizzato. Supponiamo di associare ad ogni disposizione una percentuale progressiva rispetto alle disposizioni che la precedono, e di eseguire lo stesso ragionamento sulle banche; a questo punto, iterando sulle banche, basterà individuare la sottosequenza che porta alla completa allocazione degli importi sulle banche: Se decidiamo di fermarci all'ultima disposizione con progressivo minore o uguale a quello della banca, avremmo che sicuramente riusciamo ad assegnare tutte le disposizioni, sebbene il taglio per le prime banche avverrebbe in corrispondenza di disposizioni con granularità alta, ed un conseguente aumento dell'errore sul vincolo capacitivo.
Occorre però dire che quando l'istanza è sufficientemente grande -come nell'esempio allegato alla rivista-, la misura dell'errore diviene sufficientemente piccola ed in genere si può ritenere trascurabile, soprattutto nei riguardi dei vantaggi acquisiti sui tempi di elaborazione.
Analizzando nel dettaglio le modalità di implementazione, possiamo pensare di utilizzare una self-join delle disposizioni per ottenere un array ordinato che contiene per ogni disposizione la percentuale progressiva:

```
--Creazione tabella temporanea disposizioni ordinate
CREATE TABLE #CMDTMPDispositionsQueue (
          Idx  INT   NOT  NULLIDENTITY (1, 1),
          ClientCode VARCHAR(7) NOT  NULL,
          DispositionAmount MONEY NULL
)
INSERT    INTO  #CMDTMPDispositionsQueue(ClientCode,
                                   DispositionAmount)
SELECT    T.ClientCode, SUM(ROUND(T.Amount, 2))
FROM      dbo.CMDACCTransaction T
WHERE     T.Amount > 0
AND       T.DispositionId IS NULL
GROUP     BY T.ClientCode
ORDER     BY SUM(ROUND(T.Amount, 2)) DESC

--Calcolo totale da allocare

SELECT    @TotalAmount =
                SUM(ROUND(DQ.DispositionAmount, 2))
FROM                #CMDTMPDispositionsQueue DQ
          -- Creazione tabella temporanea
          -- disposizioni con percentuale progressiva
SELECT    DQ.Idx, DQ.ClientCode,
     DQ.DispositionAmount, SUM(DP.DispositionAmount) /
          @TotalAmount * 100 AS PercentageProgressive
INTO #CMDTMPDisposition
FROM                #CMDTMPDispositionsQueue DQ
          JOIN #CMDTMPDispositionsQueue DP ON
                                  DP.Idx <= DQ.Idx
GROUP BY DQ.Idx, DQ.ClientCode,
                               DQ.DispositionAmount
```

A questo punto possiamo iterare sulle banche per trovare i punti di rottura delle sottosequenze ed assegnare le disposizioni al supporto individuato. In questo caso non siamo interessati nè agli aggiornamenti sulle banche che non saranno necessari, mentre i supporti non verranno creati preventivamente ne ad operazioni di fetch oltre la NEXT, pertanto possiamo dichiarare il cursore sulle banche FORWARD_ONLY STATIC, e per ogni banca individuare il set di eventuali disposizioni che compongono il relativo supporto e quindi creare il supporto e le disposizioni.

```
DECLARE BankAccountCursor CURSOR LOCAL
                          FORWARD_ONLY STATIC FOR
        SELECT  B.Id, B.Percentage
        FROM    dbo.CMDDATBankAccount B
        WHERE   B.Enabled <> 0
        AND     B.Percentage > 0
        ORDER   BY B.Percentage DESC
--Inizializza allocazione

SET     @CurrentPercentage = 0
SET     @DispositionsFirstIdx = 1
OPEN BankAccountCursor
FETCH NEXT FROM BankAccountCursor INTO
           @BankAccountId, @BankAccountPercentage
WHILE (@@FETCH_STATUS = 0)
        BEGIN
                SET     @CurrentPercentage =
        @CurrentPercentage + @BankAccountPercentage

--Recupera il limite superiore delle disposizioni
                SELECT  @DispositionsLastIdx =
                          ISNULL(MAX(D.Idx), 0)
                FROM    #CMDTMPDisposition D
                WHERE   D.PercentageProgressive
                          <= @CurrentPercentage

--Se ci sono disposizioni da allocare
                IF @DispositionsLastIdx >
                           @DispositionsFirstIdx
                        BEGIN
                                INSERTINTO
      dbo.CMDRIDSupport (BankAccountId, CreationDate,
         DispositionsNumber, Amount, Completed)
                SELECT
         @BankAccountId, GETDATE(), COUNT(*),
           SUM(ROUND(DispositionAmount, 2)), 1
                FROM#CMDTMPDisposition D
                WHERED.Idx
                BETWEEN @DispositionsFirstIdx AND
                           @DispositionsLastIdx
                SET @SupportId = @@IDENTITY
                INSERTINTO
                         dbo.CMDRIDDisposition (
        ClientCode, SupportId, Progressive, Amount)
                SELECT  D.ClientCode,
          @SupportId, D.Idx - @DispositionsFirstIdx + 1,
                                D.DispositionAmount
                FROM    #CMDTMPDisposition D
                WHERE
          D.Idx BETWEEN @DispositionsFirstIdx AND
                               @DispositionsLastIdx
                UPDATE  dbo.CMDACCTransaction
                SET     DispositionId = RID.Id
                FROM    dbo.CMDRIDDisposition
                                                RID
                WHERE   RID.SupportId =
                                         @SupportId
                AND
                           dbo.CMDACCTransaction.
                        ClientCode = RID.ClientCode
                AND
                dbo.CMDACCTransaction.Amount > 0
                AND
        dbo.CMDACCTransaction.DispositionId IS NULL
                        END
                SET     @DispositionsFirstIdx =
                           @DispositionsLastIdx + 1
                FETCH NEXT FROM
           BankAccountCursor INTO @BankAccountId,
                              @BankAccountPercentage
        END
CLOSE BankAccountCursor
DEALLOCATE BankAccountCursor
```

Per quanto riguarda l'errore, questo potrebbe essere ridotto se considerassimo la possibilità di effettuare due o più spazzolate sulle banche, ipotizzando inizialmente di ammettere solo gap nella soluzione temporanea, e, solo nell'ultima iterazione, completare le assegnazioni consentendo anche overlap; ovviamente le spazzolate successive alla prima verrebbero effettuate sulle disposizioni residue e sulle capacità residue, effettuando un nuovo ordinamento delle entità.

CONSIDERAZIONI FINALI

L'algoritmo è pronto per essere testato e utilizzato; per quanto riguarda il discorso dell'onerosità computazione e dell'errore sui vincoli capacitivi, occorre precisare che trattandosi di tecniche greedy è difficile prevedere l'entità di questi errori, che può variare da piccole percentuali a percentuali molto elevate nei casi degeneri.

E' interessante notare che al crescere dell'istanza, ed in particolare del numero di disposizioni rispetto al numero dei supporti, l'errore diventa sempre più piccolo, ed in genere, è preferibile avere algoritmi performanti, trattandosi sempre di soluzioni non esatte.

Carmelo Durante

JAVA CREA IL TUO BROKER DI BORSA

VI PRESENTIAMO UN METODO SEMPLICE PER REPERIRE INFORMAZIONI SUI MERCATI AZIONARI IMPAREREMO ANCHE QUALCOSA SULLA GESTIONE DEL NETWORKING E SUL PARSER DELLE PAGINE. INFINE VEDREMO COME UTILIZARE LE REGULAR EXPRESSION

Utilizzare il trading on-line sta diventando un'operazione comune. Al contrario, fino a pochi anni fa, questi strumenti erano gelosamente detenuti da una ristrettissima cerchia elitaria. I motivi che hanno permesso l'allargamento di questa platea sono sicuramente molteplici e non è questa la sede per fare un'analisi socio-economica tipo. Un aspetto di questo mutamento però ci riguarda da vicino come informatici (proprio nel senso più specifico della parola, ossia "*studiosi dell'automazione dell'informazione*"): l'avvento di Internet ha permesso di reperire informazioni ed effettuare transazioni in tempo reale, indipendentemente dallo spazio fisico che esiste fra il mercato e l'operatore. Tutto ciò inoltre può essere realizzato con dei costi irrisori, per non dire praticamente nulli.

I PROCESSI DECISIONALI

Il quadro sopra descritto può apparire tutto rose e fiori. In effetti le cose non stanno proprio così.
La grande quantità di informazioni disponibili non va di pari passo con la loro usabilità e con la loro predisposizione all'elaborazione. Facciamo un paio di esempi che possono aiutarci a focalizzare meglio ciò di cui stiamo discutendo. Immaginiamo di voler decidere se acquistare un titolo o meno. Naturalmente il solo dato che indichi la quotazione attuale dell'azione appare sicuramente troppo limitativo per prendere una decisione in merito. Sarebbe ad esempio utile disporre della serie storica del titolo stesso da una certa data in poi. Allo stesso tempo potremmo desiderare di confrontare il trend dell'azione in questione con il suo indice d'appartenenza o con un altro indice proprio per valutare la rimuneratività del settore. Infine possiamo anche applicare alcune elaborazioni a questi dati (ad esempio effettuare una media mobile) per ottenere punti significativi di "supporto" o di "resistenza". Un'altra esigenza comune potrebbe essere quella di monitorare l'andamento del nostro portafoglio di investimento rispetto alle attuali quotazioni. In altre parole, vorremo poter acquisire automaticamente le informazioni su Internet e metterle in relazione con quelle in qualche modo memorizzate off-line sul nostro computer.

METTIAMO UN PO' D'ORDINE

Ora che abbiamo dato un quadro generale della situazione, vediamo quali risposte possiamo dare alle problematiche sopra esposte. Il primo passo da compiere è stabilire come e dove reperire le informazioni che ci servono. Abbiamo parlato dei Web Service e di come essi rappresentino uno strumento per implementare applicazioni distribuite che astraggano dal linguaggio usato. In effetti se fate giro su www.xmethods.org (**box 1**) troverete vari Web Service dedicati alle quotazioni in borsa, sfortunatamente però hanno tutti come mercato di riferimento Wall Street, mentre sarebbe molto più utile avere a disposizione le quotazioni di Piazza Affari. Dobbiamo quindi rassegnarci a rinunciare ad usare un WS e trovare una soluzione meno comoda ma forse più stimolante sotto il profilo delle sfide tecniche che bisogna affrontare. Solitamente un punto di riferimento per le informazioni sulla borsa è costituito dai cosidetti "portali finanziari" che mettono a disposizione una serie di pagine web contenenti le più svariate informazioni sui mercati azionari. Naturalmente un sito web è molto funzionale quando un operatore umano naviga fra le intricate pagine del sito; lo diventa molto meno se, come nel nostro caso, vogliamo automatizzare il reperimento ed il raggruppamento dei dati. Il prosieguo di questo articolo sarà proprio dedicato a fornire una soluzione a questo problema.

FACCIAMO UN PO' D'ORDINE

Sintetizzando cosa vogliamo avere da un oggetto che si occupi di fare "il lavoro sporco per nostro

conto" possiamo affermare che abbiamo bisogno di un oggetto che ci fornisca il prezzo attuale della quotazione, la relativa data ed anche un identificativo univoco che ci permetta di riferirci senza ambiguità ad un ben determinato titolo.
La prima cosa da fare è definire un'interfaccia generica che permetta di compiere le operazioni descritte. Tale interfaccia si chiamerà *IQuoteSpy* sarà così definita:

```
public interface IQuoteSpy {
        public Date getData();
        public void setData(Date data);
        public double getPrezzo();
        public void setPrezzo(double prezzo);
        public int getId();
        public void setId(int id);
}
```

UN ROBOT PER YAHOO FINANZA

Il portale scelto è quello di Yahoo ed in particolare il sito presente all'indirizzo http://it.finance.yahoo.com/. La prima cosa che dobbiamo fare quindi è cercare di capire un po' come sia strutturato tale sito; dobbiamo cioè fare una sorta di mini reverse-engineering del sito finanziario di Yahoo.
La home page di Yahoo! Finanza è riportata in **figura 1**. La prima cosa che ci viene in mente di fare è cercare un titolo, ad esempio *Enertad* (ricordatevi di selezionare il bottone nome). A questo punto ci dovrebbe apparire la schermata di **figura 2**. Naturalmente, essendo interessati alle quotazioni di Piazza Affari, cliccheremo sul primo link, cioè sulla borsa di Milano.
Quello su cui dobbiamo concentrare la nostra attenzione è il metodo get che viene generato lato server. Una volta che abbiamo cliccato sul link veniamo indirizzati ad una pagina simile a quella riportata in **figura 3**. Osservando l'indirizzo che compare sulla relativa barra del nostro browser noteremo che è stata generata la seguente stringa:

Fig. 1: ***La home page di Yahoo! Finanza***

http://it.finance.yahoo.com/q?s= ENR.MI.
Se ora facciamo un'altra prova cercando le azioni AUTOSTRADE avremmo un indirizzo di questo tipo: http://it.finance.yahoo.com/q?s=AUTO.MI. Non ci vuole certo ora un esperto in decodifica di codici per capire come funzionano le cose: si ha una parte costante definita da http://it.finance.yahoo.com/q?s= seguito da un codice che identifica l'azione in questione ed una terminazione col suffisso .MI che fa sì che ci si riferisca alle quotazioni di Milano. Appurato ciò, collegarsi in maniera automatica alla pagina desiderata diviene estremamente semplice; è sufficiente infatti salvare da qualche parte i codici relativi alle azioni e costruire la stringa per l'url ad hoc. Veniamo

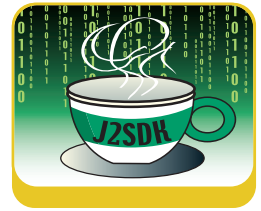



Fig. 2: ***Il risultato di un'interrogazione***

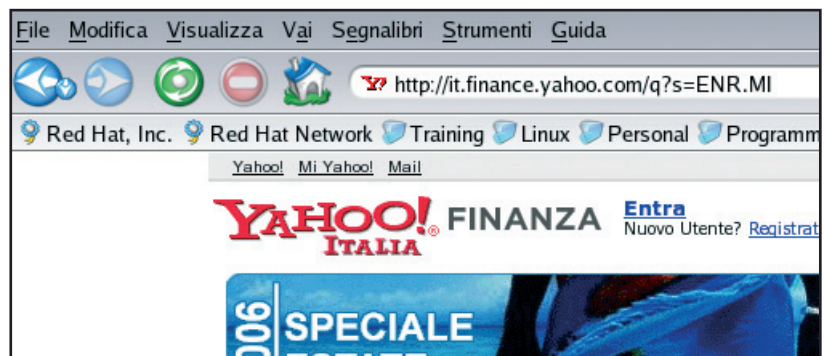
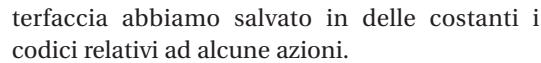



Fig. 3: ***La stringa riportata dall'interrogazione***

subito al codice che realizza quanto sopra detto. Con molta fantasia la nostra classe che implementa l'interfaccia *IQuoteSpy* si chiama Yahoo:

```
public class Yahoo implements IQuoteSpy {
...
```

Oltre che implementare i metodi previsti dall'in-

terfaccia abbiamo salvato in delle costanti i codici relativi ad alcune azioni.

```
public static final String ENI="eni";
public static final String ENEL="enel";
public static final String STM="STM";
public static final String UNICREDIT="UC";
public static final String ENGINEERING="ENG";
public static final String GENERALI="G";
.......
```

COLLEGARSI AD INTERNET IN JAVA

Vediamo quindi come ci si possa collegare ad una pagina web e leggerne il contenuto in Java. La cosa è molto più semplice di quanto si immagini, é sufficiente costruire in maniera opportuna un oggetto URL delle librerie standard di JDK ed estrarre da esso *l'OutputStream*. Nella pratica bastano queste poche righe di codice per realizzare quanto detto sopra:

```
private static final String
        link="http://it.finance.yahoo.com/q?d=v1&s=";
private static final String borsa="&m=MI";
```

queste due stringhe identificano le due parti costanti dell'URL che avevamo analizzato precedentemente, per cui, dato il codice del titolo per Yahoo!, costruiamo la stringa di connessione in questo modo:

```
String urlString
                =link+quoteName.toLowerCase()+borsa;
...
URL url = new URL(urlString);
```

Una volta creato l'URL vorremmo leggere il contenuto dell'indirizzo a cui stiamo puntando. In Java tutto ciò è molto semplice, perché grazie agli Stream possiamo astrarre dalla sorgente da cui leggiamo e quindi trattare lo Stream sempre allo stesso modo. In altre parole, per noi non cambierebbe nulla se esso provenisse da un file, da una zona di memoria, etc; questo è un concetto fondamentale ogni volta che si utilizza un linguaggio di programmazione Object Oriented. Per quanto ci riguarda basterà effettuare un wrapper sullo Stream ottenuto dall'oggetto url:

```
BufferedReader reader=null;
try {
....
                reader = new BufferedReader(new
        InputStreamReader(url.openStream()));
.....
}
```

Ora che possiamo leggere come vogliamo i contenuti della pagina web, ci dobbiamo porre due domande fondamentali: cosa otteniamo quando leggiamo una pagina? Come possiamo isolare ed estrarre le informazioni che ci interessano? Alla prima domanda è possibile dare una risposta immediata: lo stream che otteniamo dalla pagina web (e che sarà letto dall'oggetto reder creato) contiene il sorgente HTML della pagina stessa. La risposta alla seconda domanda, pur essendo collegata in parte alla prima, è subordinata alla conoscenza delle espressioni regolari.

TROVA TUTTO CON LE REGEX

Tecnicamente le espressioni regolari (RegEx = Regular Expressions) sono degli strumenti per la manipolazione di stringhe, introdotte in Java dalla versione JDK 1.4 ma già presenti in linguaggi come Python e Perl. Esse permettono di specificare dei pattern complessi di testo che possono essere trovati in una stringa di input. Le RegEx hanno una sintassi propria e per esplorarne tutte le potenzialità servirebbero svariati articoli. Quello che faremmo in questo articolo è invece presentare alcune caratteristiche contemporaneamente al loro utilizzo nella nostra applicazione. Per prima cosa quindi bisogna analizzare il sorgente HTML della pagina mostrata in **figura 3** in modo tale da costruire le RegEx appropriate per "matchare" (questo verbo sarà più chiaro tra un po') con i dati che vogliamo ricavare. Il nostro primo obbiettivo è quello di catturare la quotazione attuale del titolo. Fra le tante righe che ci appiano, con un po' di pazienza alla fine troviamo qualcosa che potrebbe fare al caso nostro:

```
</small><big><b>3,1410
                    </b></big>  ...
```

Quindi sappiamo che i prezzi sono racchiusi tra tag costanti e questo ci permette di costruire agevolmente un'espressione regolare per ricavare il valore di nostro interesse. Il seguente pattern fa quindi ciò che ci aspettiamo:

```
private static final String patternQuote =
                    "</small><big><b>([0-
                9].*)</b></big>  ";
```

È facile intuire il significato di questo pattern: esso

combacerà con tutte quelle stringhe che inizino con </small><big> seguite da una cifra qualsiasi, ossia [0-9], seguita a sua volta da un numero di caratteri qualsiasi e che termini con </big>   . Come avrete sicuramente notato la parte [0-9].* è racchiusa tra parentesi tonde. Tali parentesi non vanno considerate come parte delle RegEx, bensì esse delimitano un gruppo. Il codice seguente ci aiuta a capire meglio cosa sia un gruppo. Abbiamo creato un metodo statico che restituisce la quotazione del titolo:

```
public static double getQuote(String quoteName)
                                throws IOException{
.......
        Pattern p =
            Pattern.compile(patternQuote);
        String line;
        while((line =
            reader.readLine()) != null){
                Matcher m =
                    p.matcher(line);
                    if(m.find())
                    return
    Double.parseDouble(m.group(1).replace(",","."));
        }
```

dall'oggetto pattern ricaviamo un oggetto Matcher. Il metodo *find()* indica se la stringa di input contiene la regex che ha come pattern l'oggetto *p*. Se tale condizione è verificata catturiamo proprio la parte di stringa contenente la quotazione attuale. Questo può essere fatto con estrema semplicità proprio grazie all'utilizzo del gruppo [0-9].*. Infatti poniamo di avere una generica espressione regolare ABCD, possiamo riscriverla in maniera del tutto equivalente con A(BC(D)). In questo caso però si può accedere alla parte ABCD referenziandola con il gruppo 0, BCD con il gruppo 1, D con il gruppo 2 e così via... Nel nostro caso accediamo al gruppo 1 mediante *m.group(1)* e sostituiamo la virgola con un punto in modo tale da ottenere una stringa trasformabile in double. Arrivati a questo punto possiamo semplicemente ottenere la quotazione di un titolo con:

```
double p = Yahoo.getQuote(Yahoo.AUTOSTRADE);
        System.out.println("prezzo: "+p);
```

In una maniera del tutto simile possiamo recuperare anche la data della quotazione a cui ci riferiamo:

```
private static final String patternDate= "<small>
            ([a-z].* ([0-9].* ([a-z].*[a-z]) .*, .*)) - ";
...
        public static Date getDate(String
```

```
                    quoteName) throws IOException {
        String urlString =link+
        quoteName.toLowerCase()+borsa;
        BufferedReader reader=null;
        URL url = new URL(urlString);
        reader = new BufferedReader(new
    InputStreamReader(url.openStream()));
        Pattern p =
            Pattern.compile(patternDate);
        String line;
        while((line = reader.readLine())
                                != null){
                Matcher m =
                        p.matcher(line);
                if(m.find()){
                        String date =
  m.group(2).replace(m.group(3),""+DateUtil.convertiM
                            ese(m.group(3))).
        replace(", ","#").replace("
                    ","/").replace("#"," ");
SimpleDateFormat format = new
    SimpleDateFormat("dd/MM/yyyy HH:mm");
                        Date out=null;
                        try {
                            out
                    = format.parse(date);
                        } catch
                    (ParseException e) {
                            //
                TODO Auto-generated catch block
//e.printStackTrace();
                        }
                        return out;
                }
        }
        return new Date();
    }
```

per brevità non scenderemo nei dettagli di ogni singola riga. Come potete notare la regex necessaria a catturare la data è leggermente più complessa di quella precedente. Inoltre il mese non è espresso in forma numerica bensì con il nome in italiano, per questo è stato necessario implementare il metodo *DateUtil.convertiMese* che non fa altro che cercare su una tabella hash il numero corrispondente al nome.

```
private static final HashMap<String,Integer> mese=
                    new HashMap<String,Integer>();
    static{
        mese.put("gen",1);
        mese.put("feb",2);
        .....
    }
    public static int convertiMese(String
                                nomeMese) {
        return
```

Nel tempo che intercorre fra la stesura di questo articolo e la sua pubblicazione è accaduto che il progetto è stato accettato da Sourceforcie. Chi volesse partecipare può puntare il server http://sourceforcie.net/projects/markad

```
mese.get(nomeMese.toLowerCase().substring(0,3));
          }
```

TCP... MA QUANTO CI METTI!

Magari questi ultimi due metodi sono risultati abbastanza complessi però adesso possiamo ottenere data e prezzo della quotazione di qualsiasi titolo azionario quotato a Piazza Affari. Come abbiamo premesso all'inizio di questo articolo lo scopo è proprio quello di andare a recuperare un'ingente quantità di dati in maniera automatica per averla disponibile off-line sul nostro terminale. Se avete provato il codice fin qui riportato vi sarete accorti che l'interrogazione di una pagina web non è che sia proprio così rapida, indipendentemente dal tipo di connessione che avete a disposizione. Questo è dovuto al TCP slow start ed è un limite con il quale dovremmo fare sempre i conti. Con le linee ADSL attuali è davvero un peccato non sfruttare tutta la banda disponibile, inoltre se serializzassimo completamente la procedura, cioè se per recuperare un titolo dovessimo aspettare che quello che avevamo precedentemente chiesto sia disponibile, potremmo aspettare così tanto da rendere tutto il lavoro fatto finora poco utile in termini pratici. Una soluzione è quella di effettuare le interrogazioni su Yahoo! parallelamente, in modo tale che richiedere una quotazione o richiederne 100 impiegherà lo stesso tempo (fino alla saturazione della banda disponibile s'intende!). È quindi necessario definire un thread atomico per ogni ricerca di un titolo:

```
public class YahooThread extends Thread {
...
public  YahooThread(Yahoo yahoo,String key)  {
                    this.yahoo=yahoo;
                    this.key=key;
          ...
}

public void run() {
                    // TODO Auto-generated method
                                                  stub
                    try {
yahoo.setData(Yahoo.getDate(link));
yahoo.setPrezzo(Yahoo.getQuote(link));
                    } catch (IOException e) {
                              // TODO Auto-
                                 generated catch block
                              //e.printStackTrace();
                    }
          }
```

In pratica ogni thread non fa altro che recuperare la data ed il prezzo e valorizzare i relativi campi della classe Yahoo. Naturalmente dobbiamo però anche tenere traccia di tutti i thread che lanciamo, in modo tale da sapere quando i campi sono stati effettivamente valorizzati. A questo scopo è stata realizzata la classe YahooPool; il metodo che a noi interessa è populate, vediamo come funziona:

```
          public static void
              populate(HashMap<String,Yahoo> map) {
                    Iterator<String> iter =
                               map.keySet().iterator();
                    ArrayList<YahooThread>
                 listTh=new ArrayList<YahooThread>();
                    while (iter.hasNext()) {
                              String key = iter.next();
                              YahooThread yt = new
                       YahooThread(map.get(key),key);
                              yt.start();
                              listTh.add(yt);
                    }
                    for (YahooThread thread : listTh)
                                                        {
                              try {
                                        thread.join();
map.put(thread.getKey(),thread.getYahoo());
                              } catch
                             (InterruptedException e) {
                                        // TODO
                             Auto-generated catch block
e.printStackTrace();
                              }
                    }
          }
```

Tale metodo riceve in ingresso una HashMap che ha valorizzata solo la chiave, che identifica l'azione da cercare. Nel ciclo while vengono lanciati tanti thread quanti titoli presenti nelle chiavi. Una volta lanciato il thread esso viene memorizzato su un ArrayList. Usciti dal ciclo while cicliamo su tutti i thread lanciati ed aspettiamo che terminino per poter valorizzare in maniera corretta la HashMap ricevuta in ingresso.

CONCLUSIONI

In questo primo articolo abbiamo sviluppato il "nucleo" della nostra applicazione. Nei prossimi svilupperemo un database per memorizzare le serie storiche, grafici per visualizzare gli andamenti, metodi matematici per ricavare delle curve significative...

Andrea Galeazzi

TENERE LE REVISIONI SOTTO CONTROLLO

PARLIAMO DI SUBVERSION IL GIOIELLO OPEN SOURCE CI PERMETTERÀ DI TENERE TRACCIA DELLE MODIFICHE APPORTATE AI NOSTRI SORGENTI SENZA NEANCHE USCIRE DA VISUAL STUDIO. VEDREMO COME INSTALLARLO E USARLO

Chiunque abbia sviluppato un progetto software (anche amatoriale) avrà sentito la necessità di tenere traccia delle varie versioni di ogni singolo elemento (codice sorgente, documenti, immagini) del progetto stesso.
Perchè tale attività sia il più trasparente possibile, esistono i Version Control Systems ovvero sistemi software capaci di gestire, storicizzare ed organizzare *versioni (ovvero revisioni)* di documenti elettronici (codice sorgente di un software, documenti, immagini ecc.).
Questa è la definizione che Wikipedia fornisce del *Version Control System* (o VCS), la categoria di software a cui appartiene *Subversion* (chiamato anche SVN), il VCS di cui parleremo in questo articolo.
Sintetizzando, SVN ci permetterà di tener traccia di ogni modifica che apportiamo al nostro software (tipicamente al codice sorgente) e di gestirle in maniera tale da tornare indietro quando, per esempio, ci accorgiamo d'aver fatto un errore.
Tale strumento è ormai utilizzato dalla stragrande maggioranza dei Team di sviluppo in quanto, tra i suoi innumerevoli pregi, permette la centralizzazione dei documenti (tipicamente su un server) e la gestione remota attraverso strumenti persino integrati nel proprio IDE.

INSTALLARE SUBVERSION CON REPOSITORY LOCALE

Le componenti necessarie affinche' si possa avere un completo sistema di Versioning Control sono due:

1. La componente server ovvero Subversion installata su un server (o anche sul proprio pc) corredata da una installazione di Apache server (http server) con gli opportuni moduli.
2. Uno degli svariati client SVN da installare sui pc degli sviluppatori (nel nostro caso installeremo AnkhSVN, un plugin per Visual Studio 2005)

In questo articolo sarà considerata la sola piattaforma Windows sebbene, sia la parte server che la parte client, siano disponibili per svariate altre piattaforme (Linux, BSD ecc).
Il primo passo, dunque, sarà quello di scaricare l'ultima versione di Apache (2.0.59 nel momento in cui si scrive questo articolo) dal seguente URL: http://mirrors.publicshout.org/apache/httpd/binaries/win32/apache_2.0.59-win32-x86-no_ssl.msi e di eseguirlo per iniziarne l'installazione.
Si noti che, i moduli utilizzati da SVN sono compilati per la versione 2.0.x di Apache sebbene esista la versione 2.2.x di quest'ultimo; pertanto, in questo articolo si farà riferimento solo ad Apache 2.0.59).
Durante il setup di Apache vi verranno chieste alcune informazioni riguardanti la vostra rete ed il nome che assegnerete al server: potete fornirli basandovi sull'esempio di Figura 1.

Fig. 1: ***Setup di Apache***

Nei passi successivi vi consigliamo di lasciare le impostazioni di default e terminare l'installazione.
L'installazione sarà avvenuta con successo nel momento in cui apparirà una piuma rossa nella nostra traybar: apriamo il nostro browser preferito e digitiamo http://localhost , la pagina di default di Apache confermerà che abbiamo eseguito correttamente il setup.
Passo successivo, quindi, sara' quello di installare il pacchetto di SubVersion sul server: innanzitutto e' necessario scaricarlo dal seguente URL: http://subversion.tigris.org/ files/documents/ 15/32856/svn-1.3.2-setup.exe .
Terminato il download, lo si potrà eseguire in modo da lanciare il setup: lasceremo, così come per Apache, tutto di default, compresa l'ultima opzione che permetterà l'integrazione del server SVN con Apache.

***Fig. 2:* Setup di SubVersion**

Siamo quasi al termine del setup dell'ambiente Server: penultimo passo è la creazione del repository di SubVersion sul filesystem.
Apriamo il prompt dei comandi (Start / Esegui / cmd) e digitiamo il seguente comando:

```
mkdir c:\svn
```

che ci permette di creare una cartella principale dove verranno memorizzati i repository.
Continuiamo con il seguente comando, fondamentale per la creazione del repository stesso:

```
svnadmin create c:\svn\ioProgrammoRepository
```

Il nostro primo repository SVN è finalmente creato.
L'ultimo passo è fare in modo che Apache possa interagire con SVN
Modifichiamo il suo file di configurazione attraverso il seguente comando, digitandolo all'interno del prompt dei comandi:

```
notepad "c:\Program Files\Apache
                    Group\Apache2\conf\httpd.conf"
```

Posizioniamoci al termine del file ed aggiungiamo il seguente estratto xml

```
<Location /svn>
    DAV svn
    SVNParentPath C:\svn
    AuthType Basic
    AuthName "Subversion repositories"
    AuthUserFile passwd
    Require valid-user
</Location>
```

Tale nodo xml nel file di configurazione di Apache, informa il server web sia sul path dei nostri repository permettendone, l'accesso da tutti i pc della rete, inoltre potremo proteggere l'accesso solo ad utenti autorizzati.
Creeremo il primo utente con il seguente comando:

```
"C:\Program Files\Apache
 Group\Apache2\bin\htpasswd.exe" -c "C:\Program
              Files\Apache Group\Apache2\passwd"
                                    ioProgrammo
```

Vi verrà chiesta una password da attribuire all'utente: scegliamo *svnpass*.
I successivi utenti non avranno bisogno del flag *–c* nel comando in quanto tale opzione permette la creazione ex-novo del file che conterrà gli utenti (e le password).
Riavviamo Apache attraverso la piuma rossa (tasto destro sulla piuma /Open Apache Monitor /Restart) e, puntando il browser al seguente indirizzo http://localhost/svn/ ioProgrammoRepository , riceveremo un bel Revision 0 che ci indica il fatto che finalmente Apache parla con SubVersion ed il nostro server di Versioning è pronto!

ANKHSVN: IL PONTE CON VISUAL STUDIO

Installata la parte server, dovremmo occuparci degli sviluppatori ovvero coloro che andranno a popolare il repository appena creato con i propri progetti.
Uno dei client più diffusi per Subversion è senz'altro TortoiseSVN, un'estensione della shell di windows che cambia l'icona di una cartella o di un file in base al loro stato (modificato, nuovo, in conflitto ecc).

A volte, però, è un po' noioso e poco produttivo uscire dal proprio ambiente di sviluppo per tornare ad "Esplora Risorse" di Windows solo per inserire una modifica appena fatta.
Fortunatamente ci vengono in contro gli sviluppatori di un Add-In per Visual Studio (dalla versione 0.6 in poi anche per Visual Studio 2005) che ci permette di rimanere all'interno del nostro amato IDE.
Come sempre, il primo passo è scaricare il setup di AnhkSVN: lo sviluppo di questo addIn è ripreso da poco ed è comunque in versione Beta; il supporto per VS2005 è giovane ed è presente dalla versione 0.6 in poi.
L'URL da cui scaricarlo è il seguente: http://ankhsvn.tigris.org/files/documents/764/33676/AnkhSetup-0.6.0.2528-snapshot_35.msi .
Prima di lanciare il setup, è consigliabile chiudere tutte le istanze aperte di Visual Studio 2005.
Dopo l'installazione, aperto Visual Studio, noteremo subito una nuova voce nel menu tools: Ankh.
Di solito si configurano tre scenari tipici durante l'utilizzo di un Version Control:

1. Inserimento di un nuovo progetto nel repository
2. Caricamento, in locale, di uno o più progetti al fine di modificarli
3. Rilascio di uno o più modifiche apportate ai sorgenti del punto 2

Li analizzeremo tutti, uno alla volta.

INSERIMENTO DI UN NUOVO PROGETTO

Apriamo Visual Studio 2005 e creiamo un nuovo Progetto: nel wizard di creazione dovremo far attenzione a creare le soluzioni in modo che venga creata una cartella per ognuna di esse.
E' un accorgimento che di solito non è necessario ma Ankh lo consiglia affinche possa operare correttamente.
Per fare ciò è necessario spuntare il checkbox "Create Directory for Solution".
Fatto ciò, potete tranquillamente iniziare a scrivere il Vostro codice nella solution appena creata.
Nel momento in cui la si voglia aggiungere in SVN (magari in una nuova cartella all'interno del proprio repository), sarà sufficiente clickare col tasto destro sul nome della solution stessa all'interno del *Solution Explorer* e scegliere *"Add Solution to SubVersion Repository"*.
Come si può vedere dalla Figura3, viene presentata una finestra modale dove specificare alcune informazioni:

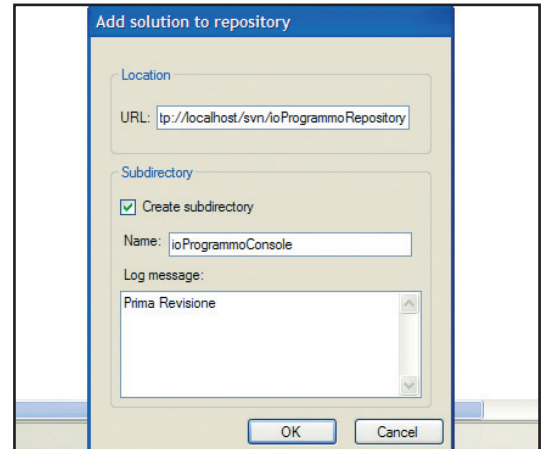



Fig. 3: Inserimento di una solution all'interno di SubVersion

URL: indirizzo del nostro repository sul server SVN. Nel nostro caso sarà *http://<nomeserver>/svn/IoProgrammoRepository*. Il *nomeserver* è l'hostname o l'indirizzo IP del server dove abbiamo installato Apache e SubVersioN.

Metteremo il check su *Create Subdirectory* che ci permette di inserire:

Name: Il nome della cartella sotto la quale vorremo inserire la nostra soluzione

Log Message: La descrizione della solution.

Dopo la conferma e l'inserimento delle credenziali dell'utente Apache creato nei paragrafi precedenti (*ioProgrammo* con password *svnpass*), ci apparirà una finestra simile a quella appena descritta: qui sceglieremo esattamente quali file del progetto inserire nel repository e, nella textarea sottostante, quale descrizione dargli. Un click su Commit spedirà il necessario sul server: et voilà avremo il nostro primo progetto sotto Controllo di Versione.

CHECKOUT E MODIFICA

Il Checkout è l'operazione che viene effettuata per trasferire uno o più documenti al fine di modificarli, tenendo traccia delle modifiche stesse.

Apriremo Visual Studio 2005 e, dal menu di Ankh sceglieremo la voce *Repository Explorer*. All'interno di questa finestra possiamo esplorare il nostro repository e fare ciò che vogliamo di ciascuna cartella al suo interno.
Clickiamo sul bottone con il segno + che ci richiede le coordinate del nostro repository (l'URL non cambia *http://<nomeserver>/svn/IoProgrammoRepository*).
Esplorando le cartelle presenti, noteremo la presenza del solo progetto appena inserito: un click col tasto destro su tale cartella ci porta in un menu contestuale molto ricco.
Noi sceglieremo semplicemente *Checkout* , specificando, quindi, la cartella locale (del nostro pc) dove tali sorgenti verranno trasferiti. A prima vista potrebbe essere un controsenso trasferire nuovamente dei sorgenti di cui si hanno le copie originali: normalmente non è così in quanto potrebbero esserci state delle modifiche effettuate da altre persone mentre noi eravamo intenti a fare altro; sempre meglio prelevare l'ultima versione disponibile. A questo punto possiamo aprire la solution appena scaricata e confermare ad Ankh il fatto di volerlo abilitare per la nostra sessione di lavoro: vedremo delle icone a forma di check verdi alla sinistra di ogni file del progetto stesso. Questo indica che tali file sono sotto il controllo di SubVersion e che noi non le abbiamo modificate (rispetto alla versione presente sul server).
Proviamo a cambiare qualcosa del Vostro progetto, magari inserendo una semplice

```
Console.WriteLine("Modificato per IoProgrammo");
```

non appena avremo salvato, l'icona accanto al file diventerà una M rossa, ad indicare che quel file è diverso dalla versione originale.

RILASCIO DELLE MODIFICHE

A questo punto, supponendo che il vostro software funzioni ancora nonostante le modifiche, si vuole mettere a sicuro i sorgenti.
Un click col tasto destro sul nome della solution all'interno del Solution Explorer ci farà apparire la voce Commit **Figura 4**.
La finestra che ci appare non ci è nuova: sceglieremo quali file modificati inviare effettivamente al server e commenteremo la modifica in modo da tenerne traccia **Figura5**.
Click su Commit e di nuovo i sorgenti saranno sul server, a disposizione di qualunque sviluppatore con un client SVN.

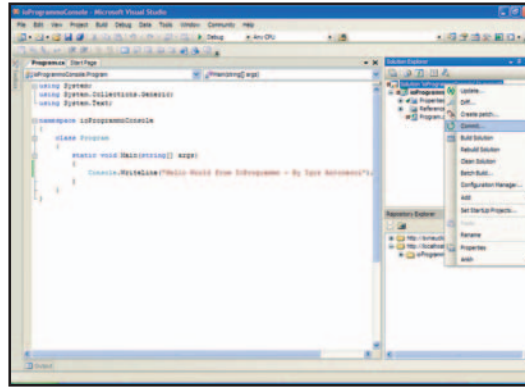



Fig. 4: ***Voce di menu per eseguire la Commit***

CONCLUSIONI

Il progetto Subversion sembrò molto ambizioso inizialmente: sostituire l'onnipresente CVS.
Sicuramente SubVersion sta guadagnando posizioni nell'ormai affollato mondo del VCS dove, però, non tutte le soluzioni sono egualmente abbordabili sia come costo che come competenze necessarie al loro utilizzo.
Le potenzialità messe a disposizione dall'accoppiata vista sono enormi. E' anche vero che SubVersion ha dalla sua uno sviluppo omogeneo portato avanti da Tigris e che il parallelo sviluppo di strumenti client come ad esempio AnkhSVN favorisce molto la diffusione di questo interessante sistema di controllo della versione. Per contro la stragrande maggioranza dei programmatori OpenSource è ancora molto legato al mondo di CVS

Antonacci Igor

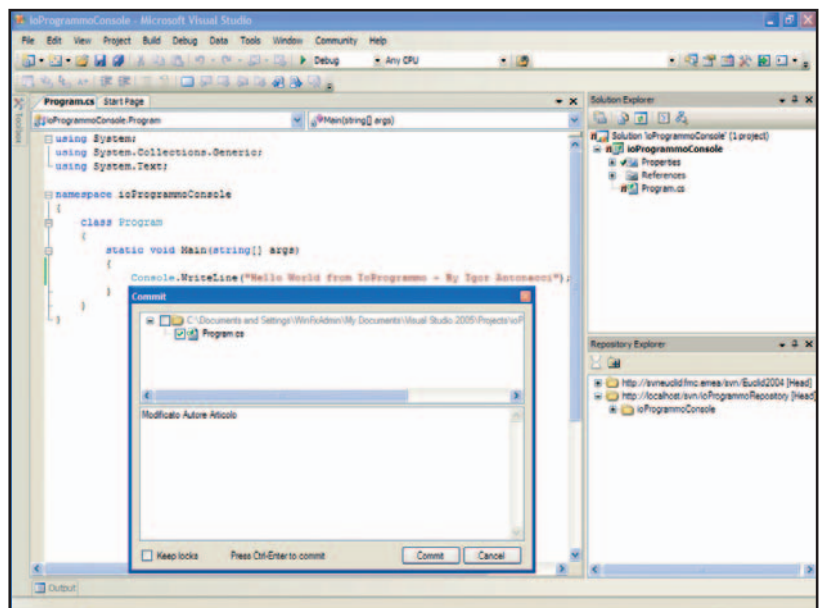



Fig. 5: ***Commit di una solution***

FATTI IL FILM DELLO SCHERMO

VOGLIAMO REALIZZARE UN VIDEO IN PRESA DIRETTA DI QUANTO ACCADE SUL MONITOR DEL NOSTRO COMPUTER. IL PUNTO È CHE VOGLIAMO ANCHE DARGLI UNO SGUARDO QUANDO SIAMO LONTANI. COME FARE? ECCO LE TECNICHE...

In questa era di internet, computer e programmi avremmo bisogno in molte situazioni di tenere sotto controllo il nostro computer. Quante volte ci capita di lasciare in esecuzione un certo programma sul nostro pc di casa e poi chiedersi quando siamo in giro "Avrà finito di….." oppure "Sarà riuscito a…..". Anche in questa situazione l'ingegno e la tecnologia può venirci in aiuto per creare un programma che fa proprio al caso nostro.

DESKTOPCONTROLLER

L'idea di base è abbastanza semplice: si vuole monitorare lo schermo del nostro desktop anche quando non siamo davanti al pc. Dobbiamo quindi strutturare la nostra applicazione con una architettura client-server. Quello quindi che avremo come riferimento per il nostro progetto viene riportato nella figura accanto.

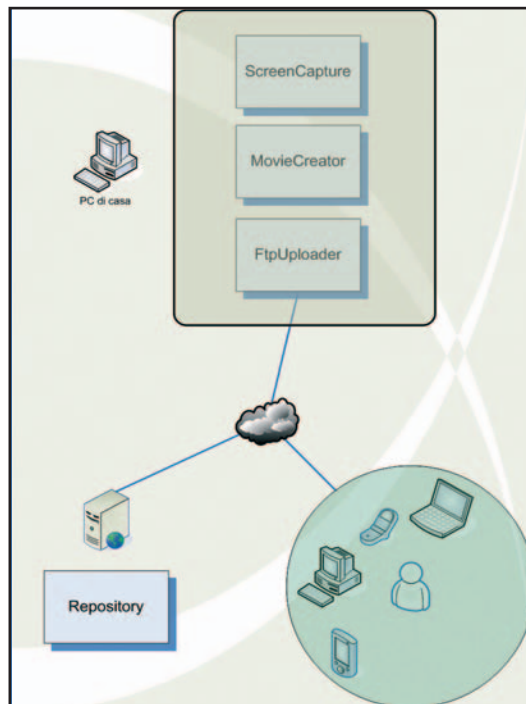



Fig. 1: Architettura del DesktopController

In questa architettura identifichiamo prima di tutto le due diverse parti dell'applicazione, ovvero il client che prepara il filmato e il server che lo visualizza. All'interno del client andremo a costruire una serie di moduli che permetto prima di tutto di catturare lo schermo, poi di trasformare le varie immagini catturate in un video. Periodicamente dovremo anche effettuare la connessione verso il server per fare l'upload tramite FTP (File Transfer Protocol) del video. Dal punto di vista del server abbiamo meno cose da fare. Dobbiamo avere una semplice pagina di autenticazione, altrimenti il desktop del nostro pc sarebbe di pubblico dominio. Dopo l'autenticazione dovremo semplicemente presentare all'utente una pagina html con il video che è stato realizzato. L'intera applicazione non è difficile da realizzare ed ora vedremo passo passo come poter scrivere tutte le componenti che ci servono.

CATTURA DELLO SCHERMO

Creare una foto istantanea dello schermo è abbastanza semplice. Java mette a disposizione la classe Robot del package java.awt, che permette in maniera molto semplice di effettuare la cattura dello schermo. Quello che noi dovremo fare di continuo è utilizzare la classe Robot e salvare l'immagine che lui ha prodotto. Il risultato della nostra applicazione non è un flusso di streaming video, ma un video che viene aggiornato ogni tanto. Per fare ciò dobbiamo pensare a quante immagini faranno parte del nostro video e soprattutto quanti "slot" avremo. Gli slot sono praticamente delle directory che utilizziamo per immagazzinare le immagini, directory che, quando avranno raggiunto il limite, verranno utilizzate da un'altra classe per creare il video. Facciamo questo perché se andassimo sempre a scrivere sui soliti file avremmo dei tempi morti nella cattura video riguardante la creazione del filmato e l'upload di quest'ultimo sul webserver. In questo modo minimizziamo i tempi morti della nostra applicazione. Per decidere quanti frame compongono il nostro

video e quanti slot abbiamo utilizzeremo un file di properties come quello riportato qui di seguito

```
frameBlocco=20
directoryCount=3
loop=ok
```

In questo modo abbiamo deciso che il video sarà composto da 20 diverse immagini e 3 directory. Il valore "loop" serve alla nostra applicazione per capire se deve continuare il funzionamento o meno. Iniziamo quindi a scrivere la classe che cattura lo schermo e lo salva su un'immagine.

```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;
import java.util.*;
import java.awt.image.renderable.*;
public class ScreenCapture {
        static int directory=0;
        static Robot robot = null;
        static Properties p = null;
        static int frameBlocco=0;
        static int directoryCount=0;
        static String loop="nonok";
        static MovieMaker mm=null;
        public static void main(String[] args) throws
                                        Exception{
                try{
                        robot = new Robot();
                        mm= new MovieMaker();
                }
                catch (Exception e) {
System.out.println("Errore creazione oggetti:
                                        "+e.toString());
        System.exit(-1);
                }
```

Dopo la definizione degli import che ci serviranno, vengono istanziate diverse variabili che verranno utilizzate nel nostro programma. Prima fra tutte "robot", l'oggetto della classe java.awt.Robot che ci permetterà di catturare lo schermo. Fra le variabili vedete anche istanziato un oggetto della classe MovieMaker, classe utilizzata per la generazione del filmato che sarà analizzata nel paragrafo successivo. Passiamo ora all'inizializzazione dei valori presi dal file di properties.

```
        try{
                        p = new Properties();
                        p.load(new
                        FileInputStream("settings.ini"));
frameBlocco=Integer.parseInt(p.getProperty("frameBl
                                        occo"));
directoryCount=Integer.parseInt(p.getProperty("direct
                                        oryCount"));
loop=p.getProperty("loop");
        }
        catch (Exception e) {
System.out.println("Errore inizializzazione proprietà:
                                        "+e.toString());
                        System.exit(-1);
        }
```

Il file che abbiamo mostrato prima è "settings.ini" ovvero il file di properties che viene caricato in questo pezzo di codice. Dopo aver caricato il file prendiamo le informazioni di cui abbiamo bisogno. Ora che sappiamo quante directory-slot avremo nel nostro programma, passiamo alla loro creazione.

```
        try{
                        for(int
                        i=0;i<directoryCount;i++) {
                                File f=new
                        File("c:\\DesktopController\\"+i);
                                f.mkdir();
                        }
        }
        catch (Exception e) {
System.out.println("Errore creazione cartelle:
                                        "+e.toString());
                        System.exit(-1);
        }
```

Abbiamo quindi preparato tutto quello di cui ha bisogno il nostro programma, quindi arriviamo direttamente al ciclo nel quale viene effettuata la cattura dello schermo.

```
while(loop.equals("ok")) {
        directory=directory%directoryCount;
        for(int i=1;i<=frameBlocco;i++) {
                cattura(i,directory);
        }
        mm.notify(directory);
        p.load(new FileInputStream("settings.ini"));
        loop=p.getProperty("loop");
        directory++;
}
}//FINE MAIN
```

Il while che abbiamo scritto continuerà a funzionare fino a quando la proprietà "loop" sul file di properties sarà settata a "ok". Quello che facciamo in questo ciclo è prima di tutto aggiornare la directory di lavoro, poi catturiamo N frame, dove N è il numero indicato nel file di properties. Quando abbiamo riempito la nostra directory di lavoro notifichiamo l'evento all'oggetto MovieMaker che si prenderà in carico il task di creazione del filmato. Il metodo cattura() richiede due parametri, il numero del frame e la directory dove posizionare questa immagine.

```
private static void cattura(int fileN,int directoryN) {
try {
        Rectangle captureArea = new Rectangle
          (Toolkit.getDefaultToolkit().getScreenSize());
        BufferedImage bufferedImage =
                robot.createScreenCapture(captureArea);
        bufferedImage =
                robot.createScreenCapture(captureArea);
        bufferedImage = ridimensiona
                                    (bufferedImage,300);
        String fileS="";
        if (fileN<10)
                fileS="00"+fileN;
        else
                fileS="0"+fileN;
        String destinazione="c:\\Desktop
          Controller\\" +directoryN+"\\"+fileS+".jpg";
        ImageIO.write(bufferedImage, "jpg", new
                                    File(destinazione));
}
catch(Exception e) {
        System.out.println("Errore nella cattura
                        dell'immagine: "+e.toString());
}
}
```

NOTA

FFmpeg è un progetto opensource utilizzato in molte applicazioni. E' disponibile per la piattaforma Linux e Windows. All'interno del progetto sono disponibili diversi eseguibili che permettono di avere un tool da linea di comando (quello che viene utilizzato nell'articolo), un server di streaming http, un semplice media player e diverse librerie per l'encoding e il decoding.
http://ffmpeg.mplayerhq.hu/projects.html

Come già anticipato il metodo cattura() utilizza l'oggetto della classe Robot per creare un'istantanea del nostro desktop. Quella che viene creata è una BufferedImage. Prima di salvarla provvediamo a ridimensionare l'immagine utilizzando il metodo ridimensiona(). Questo viene fatto solo per ottimizzare la creazione del video, ovvero per poter avere un video che si può facilmente vedere in una pagina html. Infatti se il desktop del nostro pc avesse una risoluzione altissima, avremmo poi dei problemi a vederlo su pc con risoluzione differente. Comunque, se si volesse preservare la grandezza del nostro desktop nel video, possiamo semplicemente cancellare questa riga di codice, visto che quello che viene fatto dal metodo ridimensiona() non è altro che prendere un BufferedImage e restituirne uno ridimensionato.

```
public static BufferedImage
        ridimensiona(BufferedImage source, int size) {
        int width = source.getWidth();
        int height = source.getHeight();
        float fattoreRidimensionamento = (width >
      height) ? (float) size / width : (float) size / height;
        width = (int) (source.getWidth() *
                          fattoreRidimensionamento);
        height = (int) (source.getHeight() *
                          fattoreRidimensionamento);
        BufferedImage result = new
                          BufferedImage(width, height,
                          BufferedImage.TYPE_INT_RGB);
        Graphics graphics = result.createGraphics();
        graphics.drawImage(source.getScaled
                          Instance(width, height,
Image.SCALE_AREA_AVERAGING), 0, 0, null);
        graphics.dispose(); return result;
        }
```

Abbiamo quindi analizzato la classe che ci permette di catturare immagini e di riporle in diverse directory (ridimensionandole o meno). Ora passiamo alla creazione del filmato.

CREAZIONE DEL FILMATO

Per la creazione del filmato avremmo diverse possibili soluzioni. Prima fra tutte sarebbe JMF (Java Media Framework), il framework della SUN che permette di creare e modificare flussi audio video. Un'altra possibile alternativa sarebbe quella di utilizzare la QuickTime Java API, ottime librerie che permettono di fare le stesse cose che vengono offerte da JMF, con il vantaggio di avere ulteriori potenzialità. Per la realizzazione di questo filmato è stato invece scelto un metodo più "manuale", ovvero abbiamo deciso di utilizzare un programma esterno molto noto nel mondo dell'opensource, ffmpeg (http://ffmpeg.mplayerhq.hu/). Questo progetto permette di registrare, convertire e ed effettuare stream audio video. Sono molti i progetti opensource che lo includono al loro interno e vista la semplicità d'utilizzo è stato scelto proprio questo programma, senza doverci addentrare nei meandri delle codifiche audio video di JMF e QuickTime. La classe che crea il filmato, come già precedentemente accennato, è MovieMaker.

```
import java.util.*;
import java.io.*;
public class MovieMaker extends Thread{
        int frameBlocco=0;
        int directoryCount=0;
        String loop="nonok";
        boolean toCheck=true;
        Properties p = null;
        int directory=0;
        public MovieMaker() {
        try{
                p = new Properties();
                p.load(new
                     FileInputStream("settings.ini"));
                frameBlocco=Integer.parseInt
                     (p.getProperty("frameBlocco"));
                directoryCount=Integer.parseInt
                    (p.getProperty("directoryCount"));
        }
        catch (Exception e) {
                System.out.println("Errore
            inizializzazione proprietà: "+e.toString());
```

```
                System.exit(-1);
        }
        }
        public void notify(int directory) {
                this.directory=directory;
                new Thread(this).start();
        }
```

Questa classe estende Thread perché quando viene richiamata dalla classe ScreenCapture non può bloccare la cattura delle immagini, quindi deve eseguire le sue istruzioni in maniera asincrona. Nel metodo notify(), che abbiamo precedentemente visto utilizzare nella classe ScreenCapture, viene lanciato il metodo run().

```
public void run() {
        try {
                System.out.println("Slot
                     "+directory+" completato...");
                System.out.println("Creazione del
                  filmato per directory "+directory);
                Runtime.getRuntime().exec
                          ("ffmpeg -r 5 -b 1800 -i
c:\\DesktopController\\"+directory+"\\%3d.jpg
               c:\\DesktopController\\output.mp4" );
                uploadFile("output.mp4");
        }
        catch(Exception e) {
                System.out.println("Errore nella
                    creazione video: "+e.toString());
        }
        }
}
```

Il metodo run() non fa altro che richiamare l'eseguibile di ffmpeg (che chiaramente dovrà essere inserito nel nostro path) passandogli come parametro la directory di lavoro. In questo modo lasciamo a ffmpeg il compito di generare il nostro filmato, direttamente in mp4 che è formato molto leggero.

UPLOAD DEL FILMATO

Dopo aver terminato la creazione del filmato con ffmpeg dobbiamo effettuare l'upload del file tramite protocollo FTP. Come avrete notato alla fine del metodo run() di MovieMaker richiamiamo un metodo uploadFile(). Questo metodo richiama una classe che utilizza la famosa libreria Commons Net di Apache (http://jakarta.apache.org/commons/net/index.html). Grazie a questa libreria possiamo facilmente fare l'upload del nostro filmato

```
public void uploadFile(String filename) {
                FTPUploader ftpUploader=new
            FTPUploader("mioserver","root","trout");
```

```
                ftpUploader.connect();
                ftpUploader.upload(filename);
                ftpUploader.close();
}
```

La classe FTPUploader è una classe che ci aiuta ad effettuare l'upload tramite ftp, utilizzando le librerie di Apache. Per utilizzare questa classe dobbiamo prima di tutto inizializzarla con i giusti parametri e poi richiamare in sequenza i metodi connect(), upload() e close().

```
import org.apache.commons.net.ftp.*;
import java.io.*;
public class FTPUploader {
        FTPClient ftpClient;
        String server,username,password;
        public FTPUploader(String server,String
                     username,String password) {
        this.server=server;
        this.username=username;
        this.password=password;
        }
public void connect() {
        try {
        ftpClient = new FTPClient();
                ftpClient.connect( server );
                ftpClient.login( username,
                                       password );
                System.out.println("Connesso al
                      server ftp " + server + ".");
                System.out.print(ftpClient.
                                getReplyString());
        }
        catch(Exception e) {
        System.out.println(e.toString());
        }
   }
   public void upload(String filename) {
        try {
        File f=new
           File("C:\\DesktopController\\"+filename);
          FileInputStream fis=new
                               FileInputStream(f);
          ftpClient.appendFile(filename,fis);
        }
        catch(Exception e) {
        System.out.println(e.toString());
        }
   }
   public void close() {
        try {
        ftpClient.logout();
        ftpClient.disconnect();
        }
        catch(Exception e) {
        System.out.println(e.toString());
        }
```

```
    }
}
```

Se non ci sono problemi di connessione, il video viene salvato in pochi secondi sul nostro server. Calcolate sempre che se volete automatizzare questo meccanismo ed avete un firewall, dovete dare la possibilità alla Virtual Machine di Java di poter aprire connessioni senza l'assenso esplicito, altrimenti il programma non può funzionare quando voi non siete davanti al desktop.

VISUALIZZAZIONE SUL WEB

Quello che rimane da fare è visualizzare il nostro video sul web. A questo punto del programma noi abbiamo già il video "output.mp4" sul nostro server, quindi la prima cosa che dobbiamo fare è curare almeno in parte la sicurezza. Come abbiamo già detto non possiamo far vedere il nostro desktop a tutto il mondo, quindi dobbiamo inserire una sorta di autenticazione per il nostro servizio. Per questo motivo creiamo una semplice tabella SQL sul nostro database (in questo caso MySql)

```
CREATE TABLE 'IDENTIFICAZIONE' (
        'ID' INTEGER UNSIGNED NOT NULL
                                AUTO_INCREMENT,
        'USERNAME' VARCHAR(45) NOT NULL
                                DEFAULT '',
        'PASSWORD' VARCHAR(45) NOT NULL
                                DEFAULT '',
        PRIMARY KEY('ID')
)
ENGINE = InnoDB;
```

Ora supponiamo di avere memorizzate le nostre credenziali in questa tabella, senza implementare un meccanismo di registrazione. Avremo quindi bisogno di una pagina JSP di autenticazione come la seguente

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="com.ioprogrammo.
                        desktopcontroller.db.*"%>
<html>
   <head>
      <meta http-equiv="content-type"
           content="text/html; charset=ISO-8859-1" />
         <title>Web Desktop Controller</title>
         </head>
   <body>
   <%
   String action=(String)request.getParameter("action");
   if ((action==null)||(action.equals(""))) {
   %>
         <form action='login.jsp?action=check'
                                method='POST'>
         Inserite i vostri dati:<br>
         Username: <input type='text'
              name='username' value='' size='30'><br>
         Password: <input type='password'
         name='password' value='' size='30'><br><br>
         <input type='submit' value='Login'>
         </form>
   <%
   }
   else {
         String username=(String)request.
                        getParameter("username");
      String password=(String)request.getParameter
                                ("password");
         DBCheck db;
         boolean autorizzato=false;
         String debug="";
         try {
           db=new DBCheck(username,password);
           autorizzato=db.check();
      }
      catch(Exception e) {
           out.println(e.toString());
           debug+=e.toString();
      }
         if (autorizzato) {
session.setAttribute("username",username);
         out.println("Login effettuato.<br>");
         out.println("<a href='view.jsp'>Visualizza
                                filmato</a>");
         }
         else {
              out.println("L'utente non è
         autorizzato ad accedere al servizio "+debug);
         }
   }
   %>
   </body>
</html>
```

Questa pagina JSP visualizza in un primo momento una form HTML per l'inserimento dello username e della password. Quando poi questa pagina viene richiamata con il parametro "action" uguale a "check", utilizza una classe per il controllo delle credenziali sul database. Nel caso in cui siamo in possesso di uno username e di una password per la visualizzazione del desktop remoto, allora la JSP di login inserirà il nostro username nella sessione e ci darà il link per la pagina di visualizzazione. La classe che accede al database è abbastanza semplice, visto che deve semplicemente effettuare una connessione ed inviare una query.

```
package com.ioprogrammo.desktopcontroller.db;
```

SUL WEB

Per avere maggiori approfondimenti sul web riguardanti altri modi per gestire flussi audio video con Java potete consultare i seguenti link:
JMF:
http://java.sun.com/products/java-media/jmf/
QuickTime:
http://developer.apple.com/quicktime/qtjava/

```
import java.io.*;
import java.net.*;
import java.sql.*;
public class DBCheck {
    private String username;
    private String password;
    public DBCheck(String username,String password) {
        this.username=username;
        this.password=password;
    }
    public boolean check() throws Exception {
        Class.forName("com.mysql.jdbc.Driver" );
        Connection c;
        Statement SQLStatement;
        c = DriverManager.getConnection("jdbc:mysql:
         //127.0.0.1/webdesktopcontroller", "root", "root");
        String query="SELECT ID FROM
             webdesktopcontroller.IDENTIFICAZIONE WHERE
USERNAME='"+username+"' AND
                          PASSWORD='"+password+"'";
        SQLStatement = c.createStatement();
        ResultSet rs =SQLStatement.executeQuery(query);
        int id=-1;
        while(rs.next()) {
            id=rs.getInt(1);
        }
        c.close();
        c=null;
        if (id==-1) {
            return false;
        }
        else {
            return true;
        }
    }
}
```

Siamo arrivati infine alla visualizzazione del filmato, tramite una pagina JSP che controlla la sessione ed, in caso affermativo, restituisce in output il codice HTML con i tag necessari per la visualizzazione del filmato con QuickTime. In questo caso abbiamo incluso direttamente il nome del file “output.mp4”, senza una directory, ma in un’applicazione reale, multiutente, questa pagina dovrà inserire dinamicamente la directory dell’utente con il relativo filmato

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type"
                     content="text/html; charset=UTF-8">
        <title>Web Desktop Controller</title>
    </head>
    <body>
    <h1>Web Desktop Controller</h1>
    <%
    String username=(String)session.
                    getAttribute("username");
    if ((username!=null)&&(!username.equals(""))) {
    %>
        <div align="center">
            <object classid="clsid:02BF25D5-
                8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.
                cab" height="400" width="400">
                <param name="loop"
                    value="true">
                <param name="src"
                    value="output.mp4">
                <param
                name="autoplay" value="true">
                <param
                name="controller" value="true">
                <embed height="176"
pluginspage="http://www.apple.com/quicktime/downl
    oad/" src="output.mp4" type="video/quicktime"
    width="400" controller="true" autoplay="true"
                    loop="true">
            </object><br>
            <font face="Arial,Helvetica, Geneva,
    Swiss,SunSans-Regular">Channel 1</font></div>
    <%
    }
    else {
        out.println("Utente non autorizzato!");
    }
    %>
    </body>
</html>
```

Potete ammirare il risultato nell’immagine sotto.

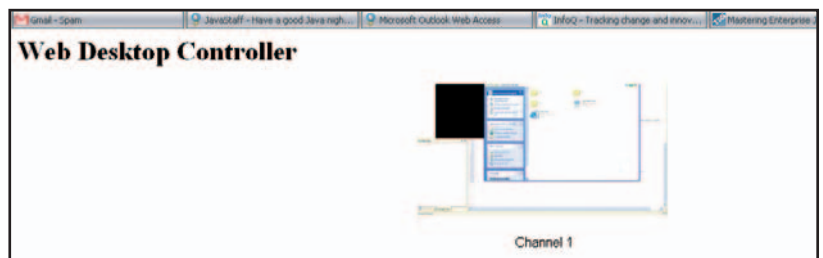



Fig. 2: ***Visualizzazione del filmato sul web***

CONCLUSIONI

L’applicazione che abbiamo illustrato in questo articolo potrebbe essere un buon punto di partenza per un servizio da offrire sul web, oppure per realizzare una sorta di controllo del desktop casalingo. Insomma, come sempre quando ci troviamo di fronte a dei programmi, i modi di utilizzarli sono molteplici e lasciati alla fantasia di chi scrive il software.

Federico Paparoni

Federico Paparoni, può essere contattato per suggerimenti o delucidazioni all’indirizzo email federico.paparoni@javastaff.com

EXIF: SCATTO DA CAMPIONE

ECCO LE TECNICHE CHE CONSENTONO DI ESTRARRE DA UNA FOTO EFFETTUATA IN DIGITALE INFORMAZIONI QUALE L'ESPOSIZIONE, LA DATA, IL NUMERO DI COLORI E, TEORICAMENTE, OGNI INFORMAZIONE CHE AVREMO VOLUTO INSERIRVI

Grazie alla versatilità e flessibilità delle macchine digitali, la fotografia sta piano piano sostituendo quella tradizionale, soprattutto in ambiti amatoriali. Un aspetto forse non molto noto è che la macchina fotografica digitale inserisce nel file jpeg della foto numerose e utili informazioni sullo scatto. Una fotocamera non è altro che un piccolo computer, che si occupa del completo controllo della fotografia. Le informazioni come data e ora di scatto, esposizione e apertura sono dunque dati immediatamente disponibili, che vengono inseriti nel file della foto. Questi prendono il nome di dati Exif. Questo è l'acronimo di Exchangeable image file format, uno standard creato dalla JEIDA (Japan Electronic Industry Development Association). Ovviamente la quantità delle informazioni presenti in un dato file variano in funzione della macchina che lo ha prodotto. Fotocamere più complesse offriranno più dati, mentre quelle più semplici ed economiche forniranno solo poche informazioni.
La struttura dei dati, detti tag, ricavata direttamente da quella usata nei file TIFF. Questa struttura copre un ampio raggio di informazioni, come:

- **Impostazioni**. Queste includono informazioni statiche come la marca e il modello della fotocamera e altri dati, come l'orientamento, l'apertura, la velocità dell'otturatore, la lunghezza focale, la modalità di determinazione dell'esposizione e la sensibilità ISO.

- **Coordinate temporali**. Le fotocamere digitali memorizzano la data e ora corrente dello scatto.

- **Coordinate geografiche**. Queste possono essere determinate da un ricevitore GPS interno alla fotocamera. I modelli che supportano questa funzionalità sono però pochi e costosi. Spesso si ovvia al problema utilizzando un dispositivo separato che memorizzi la posizione, la data e l'ora. Successivamente si farà corrispondere la posa con la posizione utilizzando come riferimento l'istante di scatto.

- **Descrizioni**. Anche questa fase viene fatta spesso in post-produzione e prevede l'associazione di testi descrittivi e di copyright all'immagine. Alcune fotocamere consentono inoltre di digitare direttamente queste informazioni, anche se l'utilizzo dei pulsanti della macchina fotografica è uno strumento poco pratico per questo scopo.

Le informazioni Exif sono importanti per tutta una serie di applicazioni. Per esempio, in fase di post-produzione, dove diviene possibile eseguire elaborazioni automatiche sulla base di tag presenti su una foto. Le informazioni di data e ora possono servire ad archiviare automaticamente i file immagine in modo organizzato nel file system.
I dati Exif in generale, se memorizzati in un database relazionale, potrebbero essere utilizzati per ricercare le foto secondo criteri diversi, come la data di scatto, la posizione geografica o eventuali parole chiave, memorizzate nei campi descrittivi.
Le informazioni Exif risultano utili anche per la realizzazione di gallerie fotografiche online e photoblog. Questi ultimi sono siti che presentano fotografie su base giornaliera, in modo similare ai blog, che invece offrono brevi testi. In queste tipologie di applicazione la possibilità di accedere a dati Exif costituisce un elemento utile per offrire informazioni sullo scatto all'utente senza la necessità di compilare queste informazioni a mano. In questo caso l'elemento principe è la data e l'ora di scatto.

Per esempio, in Figura 1 è presente un estratto da una galleria di immagini presente sul web. Sotto la fotografia è presente la data e l'ora di scatto, prelevati attraverso la funzione exif_read_data() di PHP, direttamente dal file della foto.

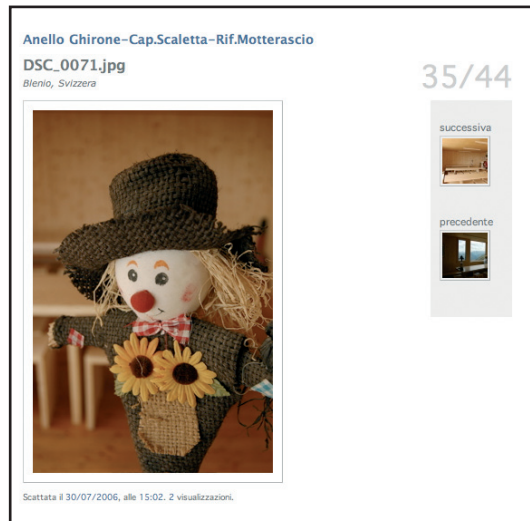



Fig. 1: ***I dati Exif possono essere impiegati dalle gallerie online per aggiungere dettagli alla foto***

MANIPOLARE I DATI

Vediamo ora come trattare i dati Exif utilizzando il linguaggio Java e alcune librerie di terze parti. La piattaforma Java infatti ad oggi non supporta i dati Exif nelle immagini jpg, sebbene queste ultime siano pienamente supportate. È possibile infatti caricare un'immagine da disco, decodificarla e visualizzarla. È possibile manipolarla eseguendo trasformazioni. È possibile generarne di nuove salvandole su file con gradi di compressione diversa. Durante tutte queste operazioni, però, se l'immagine contiene una traccia dati Exif, questa viene persa. Per esempio, se si carica un'immagine, la si ridimensiona e poi la si salva in un nuovo file, l'immagine risultante sarà priva dei dati Exif.
Questo aspetto può generare dei problemi in quei programmi che si occupano di preparare le immagini per l'upload verso una galleria di fotografie sul Web. In questo caso l'immagine inviata al sito sarà priva dei dati Exif.
Per non parlare poi del caso in cui serva elaborare i dati Exif, per esempio per scoprire la data di scatto. Le informazioni che il file system assegna all'immagine, come la data di creazione, non è significativa perché si perde a ogni invio tramite posta elettronica o con altri sistemi.
Per la piattaforma Java sono disponibili diverse librerie aggiuntive per accedere alla traccia dei dati Exif, decodificarla, scriverla e trasferirla in toto da una immagine all'altra.

LA LIBRERIA AGGIUNTIVA ZONAGEEK

Il package com.zonageek.jpeg (http://www.zonageek.com/software/java/jpeg/) è una libreria utile per leggere e manipolare file jpg con una interessante roadmap: per ora è solo in grado di elaborare la struttura del file e fornire accesso in lettura e scrittura ai dati grezzi di ciascun blocco dati Exif e IPTC, ma l'autore si prefigge di implementare la decodifica e la codifica dell'immagine e la rotazione e ritaglio senza perdita di informazioni. Purtroppo però l'ultimo aggiornamento è del 2003, quindi si presuppone che la roadmap sia stata per certi versi… abbandonata. Ed il motivo è chiaro. La libreria è nata quando l'infrastruttura di gestione delle immagini della piattaforma Java non era molto evoluta.
Per esempio, ai tempi non era disponibile **ImageIO** e quindi non disponibile alcuna funzione di base per salvare immagini in formato jpg.
Questa libreria dispone di due oggetti: **Jpg** e **ExifBlock**. Il primo rappresenta un file Jpeg, mentre il secondo un blocco dati Exif. Per leggere un file di immagine e ottenere i dati Exif è possibile scrivere:

```
Jpeg jpeg = new Jpeg();
jpeg.read(new FileInputStream(sourceFilename));
ExifBlock b = jpeg.getExifBlock();
```

Questa ultima classe contiene le definizioni dei dati supportati. Se il file contiene blocchi non censiti all'interno del codice, viene sollevata un'eccezione di tipo JpgException.
Per eseguire l'operazione opposta, e cioè scrivere un blocco Exif in un file jpg è possibile fare come segue:

```
ExifBlock b = null;
jpeg.addJpegBlock(b);
jpeg.write(new FileOutputStream(destFilename));
```

Questi metodi consentono per esempio di copiare i dati Exif da un'immagine all'altra, per esempio per creare una immagine ridimensionata di una fotografia. In questo modo da file da 3,6,8 Megapixel è possibile ottenere un file di dimensioni molto più ridotte, come per esempio 400x266. Questa versione potrà essere utile per la realizzazione di una galleria

Un'applicazione commerciale per la visualizzazione di fotografie e relativi dati Exif è ExifPro Image Viewer, scaricabile dal sito http://www.exifpro.com/.

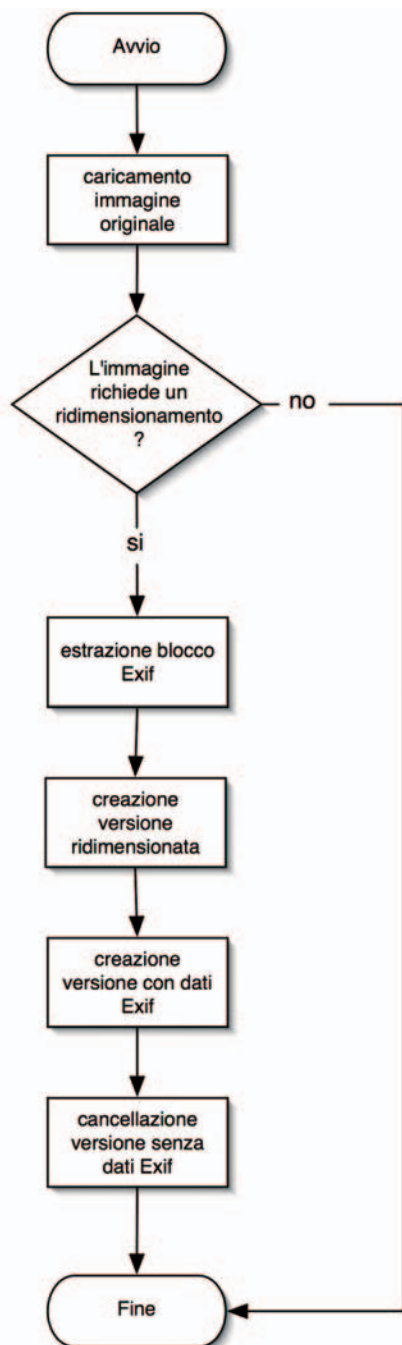



Fig. 2: algoritmo di creazione di una versione ridimensionata di una fotografia

sul Web.
L'algoritmo implementato è presentato in **Figura 2** e il codice è quello seguente:

```
public void checkDisplayImage( String
        sourceFilename ) throws IOException
{
    BufferedImage image = ImageIO.read(
            new File( sourceFilename ) );
    int width = image.getWidth();
    int height = image.getHeight();
    if (width > DISPLAY_SIZE && height >
                    DISPLAY_SIZE) {
        try {
            //prende il blocco exif
            Jpeg jpeg = new
                        Jpeg();
            jpeg.read(new
        FileInputStream(sourceFilename));
            ExifBlock b =
                jpeg.getExifBlock();
            //ridimensiona
            File tempResizedFile =
        File.createTempFile("img1", ".jpeg");
            resize(
                sourceFilename,
        tempResizedFile.getAbsolutePath(),
DISPLAY_SIZE, DISPLAY_SIZE, true );
            //copia il blocco exif
            File tempResizedWith
    ExifFile = File.createTempFile("img2", ".jpeg");
            jpeg.read(new
        FileInputStream(tempResizedFile));
            if (b != null) {
jpeg.addJpegBlock(b);
            }
            jpeg.write(new
    FileOutputStream(tempResizedWithExifFile));
            //cancella il file
            ridimensionato senza EXIF
            if
        (!tempResizedFile.delete()) {
            //gestione errore
            }
        } catch( JpegException e ) {
            e.printStackTrace();
            throw new
            IOException(e.toString());
        }
    }
    image.flush();
    image = null;
}
```

Il file con la versione ridimensionata ma completa di dati Exif avrà un nome "strano", scelto dal sistema attraverso il metodo **File.create TempFile()**. Come passo finale è dunque possibile rinominare il file in modo che assuma un nome più significativo. Per esempio, se il nome originale è IMG_9828.jpeg è possibile far assumere al file ridimensionato il nome IMG_9828_small.jpeg.

ALL'INTERNO DI ZONEGEEK

La libreria ZoneGeek si compone di una manciata di classi, la maggior parte delle quali è relativa alla gestione di blocchi di dati perso-

nalizzati. Sono presenti dunque le classi AdobeBlock, ExifBlock o JfifBlock.
Una caratteristica da segnalare di questa libreria è il fatto che se un dato Exif non viene riconosciuto, viene sollevata una eccezione. Questo significa che l'elaborazione si interrompe, anche se l'unico intento del programmatore è quello di copiare i dati Exif da un file all'altro. Per ovviare a questo problema è necessario modificare il codice della libreria, che per fortuna è open source e quindi disponibile in formato sorgente. In particolare, per ottenere il funzionamento dell'algoritmo descritto sopra è necessario modificare la classe **ExifBlock**, e nel metodo **readIFD**() commentare il sollevamento dell'eccezione, sostituendolo con un semplice **System.out.println**():

```
//throw new JpegException("Unexpected data,
//unknwon EXIF tag "
//          + tag + " (" + mode + ") " + data);
System.out.println("Unexpected data, unknwon
                                        EXIF tag "
          + tag + " (" + mode + ") " + data);
```

Inoltre, per alcuni elementi informativi, sembra che i file di prova utilizzati per provare la libreria contenessero dati non validi. In particolare, alcuni elementi di tipo SHORT contenevano in realtà dati di altra dimensione, facendo fallire la conversione di tipo seguente:

```
this.putShort(itemPos,
                    ((Short)value).shortValue(),
                                this.isBigEndian);
```

per ovviare a questo inconveniente l'operazione è stata racchiusa da un blocco try/catch che sopprimesse l'errore con una semplice segnalazione a console:

```
case ExifTagInfo.TYPE_SHORT:
        // TODO: Check sign
        if (itemCount == 1) {
          try {
            this.putShort(itemPos,
                    ((Short)value).shortValue(),
            this.isBigEndian);
          } catch( ClassCastException e ) {
                  //e.printStackTrace();
                  System.out.println(
                                        "writeIFD()
                                  EXIF scartato " +
value.getClass().getName() + "=" + value );
          }
        }
```

In questo caso si disporrà di un dato in meno, ma almeno l'operazione di parsing nella sua globalità viene completata. La classe JpegBlock e le relative sottoclassi dispongono di numerosi metodi getter e setter che permettono di estrarre e impostare i diversi elementi informativi. Per enumerare le informazioni disponibili in un file di immagine è possibile utilizzare il seguente algoritmo, mutuato dal codice sorgente della libreria. Per prima cosa viene aperto un file, da cui vengono estratti i blocchi dati utilizzando il metodo getBlocks(). Per ciascuno di questi viene estratta una Map che contiene le informazioni presenti, rappresentate come coppia chiave/valore:

```
Jpeg jpeg = new Jpeg();
jpeg.read(new FileInputStream(inFileName));
Vector blocks = jpeg.getBlocks();
for (int i = 0; i < blocks.size(); i++) {
        JpegBlock block = (JpegBlock)
                                blocks.elementAt(i);
        Map info = block.getInformation();
        System.out.println("Blocco originale alla
                              posizione " + i + ": ");
        System.out.println("  Marcatore " +
                              info.get("Jpeg Block"));
        for (Iterator keys =
      info.keySet().iterator(); keys.hasNext(); ) {
                Object key = keys.next();
                System.out.println("  - " + key
                  + ": [" + info.get(key) + "]");
        }
}
```

Questo è l'output prodotto eseguendo questo programma e fornendo in input una fotografia scattata con una Nikon D50. Si può notare come nel blocco Exif APP1 sia presente il tempo di esposizione (ExposureTime, 10/2000), la lunghezza focale (FocalLength, 55), l'orientamento (Orientation, 1), l'apertura (MaxApertureValue, 5) e la data e ora di scatto (Datetime, 2006:07:30 11:20:55):

```
Blocco originale alla posizione 0:
  Marcatore EXIF (APP1)
```

WINDOWS XP ED EXIF

Windows XP dispone di una funzionalità di base che supporta i dati Exif. Questi sono visualizzati in una tabella nelle proprietà del file, quando viene selezionato un file che le contiene. In alcuni casi però questa operazione può corrompere la sezione Exif del file.

```
  - ExposureTime: [{num=10, val=0.0050,
                                    den=2000}]
  - TIFFYCbCrPositioning: [2]
  - EXIFVersion: [0221]
[...]
Blocco originale alla posizione 3:
  Marcatore DHT
  - Jpeg Block Length: [416 bytes]
  - Jpeg Block: [DHT]
Blocco originale alla posizione 4:
  Marcatore SOS
  - Jpeg Block Length: [2486881 bytes]
  - Jpeg Block: [SOS]
```

UN'ALTERNATIVA

La libreria illustrata è compatta e veloce da usare, ma non è il massimo dell'aggiornamento. È disponibile anche la libreria jpeg-exif di dreawnoakes.com (http://www.drewnoakes.com/ code/exif/). Questo software, nato come programma di utilità per estrarre la data di scatto dalle fotografie si è evoluto in un framework completo per operare con dati Exif e Iptc. Il package com.drew.metadata.exif supporta metadati specifici per diversi modelli di Nikon, Canon, Casio, Olympus, Kodak, Kyocera, Panasonic, Pentax e Sony.
L'utilizzo di questa libreria è similare al precedente. Il seguente listato produce l'elenco dei metadati presenti in un file jpeg. Come si nota, l'integrazione con la piattaforma Java è migliore. Tramite il metodo readMetadata() della classe JpegMetadataReader si ottiene un oggetto Metadata. Da questo si può ottenere un iteratore che ritorna un insieme di oggetti Directory. Ciascuno di questi dispone di un elenco di elementi Tag:

```
File jpegFile = new File("myImage.jpg");
Metadata metadata =
       JpegMetadataReader.readMetadata(jpegFile);
Iterator directories =
                     metadata.getDirectoryIterator();
while (directories.hasNext()) {
    Directory directory =
                     (Directory)directories.next();
    Iterator tags = directory.getTagIterator();
    while (tags.hasNext()) {
        Tag tag = (Tag)tags.next();
        System.out.println(tag);
    }
}
```

NOTA

JAlbum (http://jalbum.net/) è una eccellente applicazione per la generazione di album di fotografie online. Il programma è scritto in Java e quindi multipiattaforma.

La classe Tag contiene il singolo elemento informativo e offre metodi getter per ottenere il tipo, il nome e la descrizione del dato. Opportuni metodi di conversione, presenti nella classe Directory, consentono di ottenere il valore del tag sottoforma di intero, double, float, long, numero razionale o data.
Un esempio di output del programma precedente, sempre relativo a una fotografia scattata con una macchina fotografica Nikon, è il seguente:

```
[Exif] Image Description =
[Exif] Make = NIKON
[Exif] Model = E990
[Exif] Orientation = top, left side
[Exif] X Resolution = 1/300 inches
[Exif] Y Resolution = 1/300 inches
[Exif] Resolution Unit = Inches
[Exif] ISO Speed Ratings = 100
[Exif] Exif Version = 2.10
[Exif] Date/Time Original = 2000:12:30 10:18:16
[Exif] Date/Time Digitized = 2000:12:30
                                       10:18:16
[Exif] Focal Length = 8.2 mm
[Exif] User Comment =
[Exif] FlashPix Version = 1.00
[Exif] Color Space = sRGB
[Exif] Exif Image Width = 2048 pixels
[Exif] Exif Image Height = 1536 pixels
[Nikon Makernote] Makernote Unknown 1 =
[Nikon Makernote] ISO Setting = Unknown (0 0)
[Nikon Makernote] Color Mode = COLOR
[Nikon Makernote] Quality = FINE
```

CONCLUSIONI

Attraverso Java e le due librerie qui descritte è possibile andare oltre nella gestione della propria libreria di fotografie. Tramite la lettura dei metadati presenti nei file è possibile implementare molte applicazioni simpatiche, come gallerie di foto, o motori di ricerca specifici.
I dati Exif rappresentano una risorsa fino ad ora abbastanza trascurata ma che se utilizzata correttamente potrebbe rappresentare la soluzione a diversi problemi legati alla gestione delle immagini

Massimiliano Bigatti

PROGETTI CONCRETI

BlueMarine (http://bluemarine.tidalwave.it/) è un interessante progetto italiano che emula applicazioni come Aperture di Apple o Lightroom di Adobe per oranizzare, sviluppare, stampare e pubblicare le foto digitali.

IBM INVENTA IL DATABASE IBRIDO

AL GIORNO D'OGGI TUTTO È XML. ALLORA PERCHÈ SALVARE I DATI IN UN FORMATO NATIVO PER POI DOVERLI ELABORARE? VALE LA PENA CREARE UN DATABASE CHE GESTISCA DIRETTAMENTE IL FORMATO XML. DB2 VIPER FA PROPRIO QUESTO, OLTRE TUTTO IL RESTO...

Forse non tutti sanno che la gestione dei database compie 40 anni. Infatti, negli Anni 60, quando l'allora presidente degli Stati Uniti, John F. Kennedy diede il via ai viaggio sulla Luna, nacque il problema di dove memorizzare grandi quantità di dati relativi alla missione Apollo. Quindi è nel 1966 che nasce l'*Information Management System, IMS*, il sistema per gestire l'informazione, in una parola: il database. Ed è stata IBM a realizzarlo. Un'altra pietra miliare dell'informatica è stata l'invenzione del modello relazionale e del linguaggio *SQL, Structured Query Language* intrdotto, anche in questo caso da IBM. Oggi, ancora IBM vuole segnare un altro momento storico per il database: la nascita del db ibrido, sia relazionale sia xml. Da più parti si sente parlare di interoperabilità tra sistemi eterogenei e dei web services come il mezzo per realizzarla, attraverso scambio di informazioni in formato xml. In questo contesto, un db che memorizzi le informazioni sia nel modo tradizionale sia in xml può davvero segnare una svolta nello sviluppo dei db. Se a questo si aggiunge anche che è gratuito, ecco che si hanno tutte le caratteristiche di IBM DB2 versione 9, nome in codice Viper. L'obiettivo è adesso quello di capire come usarlo subito, come creare tabelle, leggere, inserire o cancellare dati, creare indici e lanciare stored procedure, usando il db ibrido e gli strumenti di sviluppo e amministrazione di IBM.

Fig. 1: Database relazionale con xml

LETTURA DEL FORMATO XML

Per prendere confidenza gradualmente con i dati in formato xml e per vedere come Java gestisce le query, il file migliore è *XmlRead*. Con gli esempi contenuti in questa classe, chiunque può utilizzare da subito sia DB2 Viper sia le strutture di Java per accedervi. I metodi principali sono 5: i primi 3 permettono di leggere dati dal db usando le varie api java; gli altri 2 accedono ai valori xml in formato *Clob* e *Blob*. Il modo più semplice per realizzare una query si trova in *execQuery* e le righe più importanti sono le seguenti.

```
// XmlRead -> execQuery
ResultSet rs = stmt
   .executeQuery(" SELECT cid,
               XMLSERIALIZE(info as varchar(600)) "
   + "FROM customer WHERE cid < 1005 ");
...
while (rs.next()) {
   customerid = rs.getInt(1);
   customerInfo = rs.getString(2);
}...
```

Viene eseguita la query e poi si prendono i dati. Tutto molto semplice e standard, con l'aggiunta dell'istruzione *XMLSERIALIZE* che converte xml in stringa o in oggetti *Blob*.
Gli altri metodi per eseguire le query sfruttano il *PreparedStatement* di Java che permette di memorizzare le stesse, ottenendo prestazioni migliori.

```
// XmlRead -> execPreparedQuery
PreparedStatement pstmt = con
```

```
  .prepareStatement(" SELECT cid,
            XMLSERIALIZE(info as varchar(600)) "
  + " FROM customer WHERE cid < 1005 ");
// XmlRead -> execPreparedQueryWithParam
PreparedStatement pstmt = con
  .prepareStatement(" SELECT cid,
            XMLSERIALIZE(info as varchar(600))"
  + " FROM customer WHERE cid < ? ");
pstmt.setInt(1, 1005);
```

In questi 2 metodi la query viene memorizzata. Nel secondo caso, poi, c'è la possibilità di passargli ogni volta un parametro diverso. Quest'ultimo caso è abbastanza comune se si pensa che, durante l'esecuzione del programma, la query rimane la stessa, mentre cambiano i parametri con la quale viene interrogata. La maggior parte dei database tratta il formato xml come una grande quantità di dati, al pari di file audio o video o immagini senza avere la possibilità di manipolarli. Anche DB2 Viper può farlo, sebbene la sua caratteristica sia quella di andare oltre questa gestione.

Fig. 2: ***Dato in formato relazionale.***

Fig. 3: ***Dato in formato xml***

CARATTERISTICHE DB2 EXPRESS-C

Tra le varie versioni del DB2 Viper, una in particolare è rivolta agli sviluppatori e al mondo accademico ed è la versione Express-C che è gratuita. Quello che è escluso da questa versione è il Warehouse, il supporto per la connessione, gli strumenti web. Non ci sono però limitazioni sulla quantità di dati memorizzabili.

```
// XmlRead -> ReadClobData
ResultSet rs = stmt
  .executeQuery("SELECT cid, XMLSERIALIZE
                          (info as Clob)"
  + " from customer where cid < 1005");
...
while (rs.next()) {
  customerid = rs.getInt(1);
  Clob clob = rs.getClob(2);
  long clob_length = clob.length();
  temp_clob = clob.getSubString(1, (int)
                          clob_length);
}
// XmlRead -> ReadBolbData
PreparedStatement pstmt = con
  .prepareStatement("  SELECT cid,
              XMLSERIALIZE(info as blob)"
  + "  from customer where cid < 1005 ");
...
while (rs.next()) {
  customerid = rs.getInt(1);
  Blob blob = rs.getBlob(2);
  byte[] Array = blob.getBytes(1, (int)
                          blob.length());
  String temp_string = "";
  for (int i = 0; i < Array.length; i++) {
    char temp_char;
    temp_char = (char) Array[i];
    temp_string += temp_char;
  }
}
```

Blob e *Clob* sono oggetti che immagazzinano dati di grandi dimensioni memorizzati come stringhe binarie o di caratteri.
Questa prima classe serve per usare da subito tutte le funzionalità del db con Java. Per sfruttare al meglio DB2 Viper si può vedere la classe successiva.

INSERIMENTO DI VALORI XML

Se per leggere dei valori xml, l'unico problema è come trattarli per formattarli in modo corretto (come stringhe, *Clob* o *Blob*), per l'in-

serimento emerge prepotentemente il lavoro fatto dagli sviluppatori per garantire la correttezza dei dati. Infatti, il modello di dati relazionale usa i vincoli di chiave, come la chiave primaria o quella esterna, per verificare che i dati inseriti siano coerenti con quelli esistenti. Ci sono anche i vincoli come il *check* che permette di dire che un certo campo deve assumere determinati valori. Inoltre dovrebbe essere possibile selezionare solo parte dei dati xml e non tutto il file in blocco. Tutti questi controlli dovrebbero essere validi anche per i dati xml ed è questa la caratteristica innovativa del DB2 Viper. Il file *XmlInsert* contiene molti esempi su come eseguire l'inserimento con controlli e validazioni sui dati. Questa volta gli esempi più semplici vengono tralasciati per entrare direttamente nel cuore della tecnologia. Il primo metodo inserisce una riga dopo aver preso le informazioni dalla riga di un'altra tabella e averle trasformate in xml.

```
// XmlInsert -> InsertwhereSourceisXmlFunction
Statement stmt1 = con.createStatement();
stmt1.executeUpdate("INSERT INTO
                                    customer(cid,info) "
+ "SELECT ocid, XMLPARSE(document
                               XMLSERIALIZE(content "
+ "XMLELEMENT(NAME\"oldCustomer\",
                                      XMLATTRIBUTES"
+ "(s.ocid,s.firstname||' '||s.lastname AS
                                            \"name\")) "
+ "as varchar(200)) strip whitespace) "
+ "FROM oldcustomer s " + "WHERE
                                        s.ocid=1010");
```

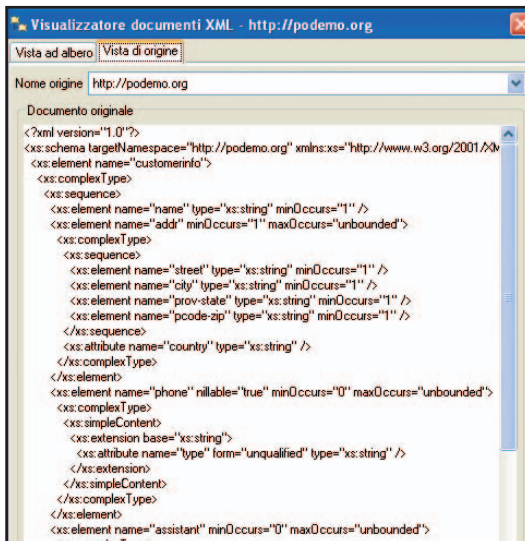



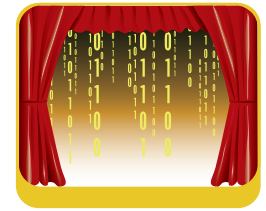
***Fig. 4:* Schema per customer**

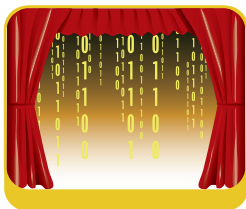
I dati sono presi dalla tabella *oldcustomer*, manipolati usando le funzioni xml e inseriti come campo xml nella tabella *customer*. In particolare, il nuovo dato avrà nel campo xml della tabella *customer*, dentro l'attributo *name*, il valore nome e cognome che invece, nella tabella *oldcustomer*, è memorizzato in due campi relazionali distinti. Una foto del passaggio chiarisce meglio la trasformazione da dati dal formato relazionale (db classico con tabelle e campi) in formato xml con variazione dei dati.

Anche in questo esempio c'è l'uso di funzioni *SQL/XML* che servono per eseguire le trasformazioni sui dati. In particolare, *XMLELEMENT* serve per convertire i dati memorizzati nel tradizionale formato di colonna, in un elemento xml. Discorso analogo vale per *XMLATTRIBUTES* per la conversione in un attributo. *XMLPARSE*, invece, è la funzione che converte una stringa in formato xml.

Sempre in tema di trasformazioni, dopo essere passati da relazionale a xml, si potrebbe passare da file xml a db xml. Il tutto usando come mezzo di trasformazione dei dati l'oggetto *Blob*.

```
// XmlInsert -> InsertwhereSourceisBlob
String xsdData = new String();
xsdData = returnFileValues("expPrg.xml");
byte[] byteArray = xsdData.getBytes();
// Create a BLOB object
java.sql.Blob blobData =
               com.ibm.db2.jcc.t2zos.DB2LobFactory
   .createBlob(byteArray);
PreparedStatement pstmt = con
   .prepareStatement("INSERT INTO
                                  customer(cid,info) "
      + "VALUES(1021,XMLPARSE
   (document cast(? as Blob) strip whitespace))");
pstmt.setBlob(1, blobData);
pstmt.execute();
```

Dal codice si apprezza meglio il passaggio file xml -> blob -> db xml. In particolare si legge il dato dal file expPrg.xml, lo si trasforma nell'oggetto *java.sql.Blob* che quindi è nativo di java e non fa parte di api nuove, e infine si

QUANTO È PIÙ VELOCE?

Per dimostrare i miglioramenti introdotti con DB2 Viper rispetto ai tradizionali db, si sono calcolate alcune statistiche. Creazione di un nuovo report impiega 10 minuti, anziché 1 giorno. Selezione delle porzioni di dati di interesse nei web services riduce il tempo di lettura e scrittura del 65%.
Il cambiamento dello schema richiede pochi minuti, invece di un giorno interno. Implementare il recupero di dati da xml impiega 30 minuti, recupero da Clob 8 ore.

esegue l'inserimento nel db in formato xml. Come si vede dal codice, c'è l'istruzione *XMLPARSE* che serve, come suggerisce il nome, a fare il parse, la trasformazione, in formato xml. In realtà, poiché il campo di destinazione è in formato xml, l'istruzione *XMLPARSE* non è necessaria, e può essere eseguita implicitamente come mostra il metodo *InsertBlobDataWithImplicitParsing*.

```
// XmlInsert -> InsertBlobDataWithImplicitParsing
PreparedStatement pstmt = con
    .prepareStatement("INSERT INTO
                              customer(cid,info) "
    + "VALUES(1022, cast(? as Blob))");
```

Il passo successivo è quello di usare il *Clob* come mezzo di trasformazione da file xml a db e questo viene fatto nel metodo *Insert whereSourceisClob*.
Dopo aver visto i metodi che manipolano i dati in formato xml, si può passare a quelli che considerano la struttura xml e la correttezza dei valori. Il primo metodo è *InsertFrom StringNotWellFormedXML*. Il nome già suggerisce che c'è qualche problema. L'errore sta nel codice seguente e magari qualcuno riesce a trovarlo senza leggere la soluzione!

```
// XmlInsert ->
                    InsertFromStringNotWellFormedXML
PreparedStatement pstmt = con
    .prepareStatement("INSERT INTO
                    customer(cid,info) VALUES(1032,"
    "'<customerinfo
       Cid=\"1032\"><name>Romina</name>')");
```

Chi è abituato a lavorare con i file xml, forse ha notato che manca il tag di chiusura di *cutomerinfo*, cioè *</customerinfo>*, come mostrato in seguito.

```
PreparedStatement pstmt = con
    .prepareStatement("INSERT INTO
                    customer(cid,info) VALUES(1032,"
    "'<customerinfo
    Cid=\"1032\"><name>Romina</name></custo
                                  merinfo>')");
```

Al di là del fatto che qualcuno abbia notato l'errore, la cosa innovativa è che il DB2 Viper si è accorto che il formato xml che si tenta di inserire ha una struttura errata e lo segnala lanciando un'apposita eccezione.
Ancora a caccia di errori con i due metodi seguenti. In questi casi c'è la validazione dello schema xml. Lo schema serve per dare una struttura precisa ai dati che verranno inseriti.

Nella figura si può apprezzare lo schema per l'identificativo *customer*. Poiché i file xml sono composti da elementi e attributi, lo schema definirà quali di questi devono essere obbligatori. Si può notare come l'elemento *addr* deve avere almeno un'occorrenza e che è composto dagli elementi *street*, *city*, *prov-state* e *pcode-zip* anche loro obbligatori. Il metodo *ValidateXML Document* riesce ad inserire la riga correttamente nel db, mentre quello *InsertwithValidation Sourceis Varchar* non lo fa.

```
// XmlInsert -> ValidateXMLDocument
stmt.executeUpdate("INSERT INTO
                              customer(cid,info) "
    + "VALUES(1037, "
    + "XMLVALIDATE( "
    + "XMLPARSE(document '"
    + "<customerinfo xmlns=\"http://podemo.org\"
                                              "
    + "Cid=\"1037\">"
    + "<name>Maurizio</name>"
    + "<addr country=\"italy\">"
    + "<street>via campioni, 4</street>"
    + "<city>Roma</city>"
    + "<prov-state>Roma</prov-state>"
    + "<pcode-zip>00100</pcode-zip>"
    + "</addr>"
    + "</customerinfo>' preserve whitespace)"
    + "according to XMLSCHEMA ID
                                    CUSTOMER))");
```

Come detto, con il codice sopra si riesce ad inserire la riga. Invece il metodo seguente fa riferimento a un'altra riga presente nel db che fa lanciare un'eccezione relativa allo schema xml usato.

```
// XmlInsert ->
               InsertwithValidationSourceisVarchar
stmt1.executeUpdate("INSERT INTO
                              customer(cid,info) "
    + "SELECT ocid, XMLVALIDATE ( XMLPARSE(
                                  document addr "
    + "preserve whitespace ) according to "
    + "XMLSCHEMA ID CUSTOMER) "
    + "FROM oldcustomer p "
    + "WHERE p.ocid=1009");
```

Il codice sopra, che è corretto, fa riferimento al campo *addr* della tabella oldcutomer dove c'è il valore xml che da errore.

```
// tabella oldcusotmer, riga con ocid = 1009
<customerinfo xmlns="http://podemo.org"
                                     Cid="1009">
<addr country="Italy">
```

```
<street>via campioni, 4</street>
<city>Rome</city>
<prov-state>Rome</prov-state>
<pcode-zip>00100</pcode-zip>
</addr>
<phone type="work">06-91132776</phone>
</customerinfo>
```

Per capire dove si è insinuato l'errore, si devono considerare 3 elementi: 1) lo schema di riferimento riportato in figura; 2) la struttura xml sopra; 3) la struttura xml del metodo *ValidateXMLDocument* che aveva dato esito positivo. Le differenze tra le 2 strutture, non sono solo nel campo in più phone. Questo campo, infatti, riguardando la struttura in foto, è ammesso con *minOccurs="0"* e quindi non è obbligatorio. Diverso il discorso per l'elemento name, che è presente nel primo caso e non nel secondo e che, secondo lo schema, ha *minOccurs="1"* e quindi è obbligatorio.
E con questi due metodi si hanno a disposizione altre righe di codice per inserire e validare gli schemi prendendo come input i dati da tabella o mettendoli nel codice.
Generalmente con un db si eseguono operazioni di inserimento, modifica e cancellazione dei dati: dopo aver visto l'inserimento, si può passare alla modifica.

MODIFICA DI VALORI XML

Anche per modificare i dati si possono presentare svariati casi a seconda di quale input si deve maneggiare.
Nel file *XmlUpDel* ci sono tutti i metodi per eseguire le modifiche. Tanto per definire bene il contesto nel quale si sta lavorando, il primo metodo permette di eseguire l'aggiornamento nel modo più semplice.

```
// XmlUpDel ->
                   mostSimpleUpdatewithConstantString
PreparedStatement stmt1 = con
   .prepareStatement("UPDATE customer"
   " SET
     info=XMLPARSE(document'<newcustomerinfo>
                                        <name>Max"
   + "<street>park street</street>"
   +
"<city>delhi</city></name></newcustomerinfo>
                                           'preserve "
   + "whitespace) WHERE cid=1008");
```

\Questo metodo rappresenta quasi la normale operazione di *update* unita al parse del documento xml, come già detto negli esempi precedenti. Casi molto simili a questo prendono come input i dati da una colonna in formato stringa o da una colonna in formato xml.
Per analogia con l'inserimento, un caso interessante è l'aggiornamento dei dati in cui la sorgente viene trasformata in *Blob* ed effettuato il *parse* implicito.

```
// XmlUpDel ->
        UpdatewhereSourceisBlobWithImplicitParsing
String xsdData = new String();
   xsdData = returnFileValues("expPrg.xml");
   byte[] Array = xsdData.getBytes();
java.sql.Blob blobData =
  com.ibm.db2.jcc.t2zos.DB2LobFactory.createBlob(
                                          Array);
PreparedStatement pstmt = con
   .prepareStatement(" UPDATE customer SET
                                          INFO = "
      + " cast(? as Blob)" + " WHERE cid=1008");
pstmt.setBlob(1, blobData);
pstmt.execute();
```

Questo caso è davvero quasi un ripasso: si legge il dato dal file *expPrg.xml* lo si trasforma in *Blob* e si esegue l'*update*. Anche in questo metodo il parse viene eseguito in maniera implicita non essendo presente l'istruzione *XMLPARSE*. Infine resta il caso della cancellazione.

CANCELLAZIONE DI VALORI XML

Per completezza c'è il caso della cancellazione sia di una riga che di una tabella.

```
// XmlUpDel -> DeleteofRowwithXmlData
PreparedStatement stmt1 = con
   .prepareStatement("DELETE FROM customer"
      + " WHERE cid=1008");
stmt1.execute();
```

Con questo metodo la riga relative all'identificativo 1008 è cancellata. Per eliminare un intera tabella c'è invece il metodo rollbackChanges.

SQL/XML FUNZIONALITÀ DEL 2006

Alcune novità di SQL/XML sono le funzioni: XMLQUERY, per eseguire una XQuery; XMLESISTS, per determinare se una query ha un risultato di ritorno; XMLTABLE, per eseguire una XQuery e ottenere come risultato una tabella relazionale.

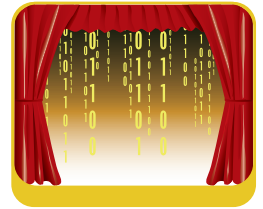
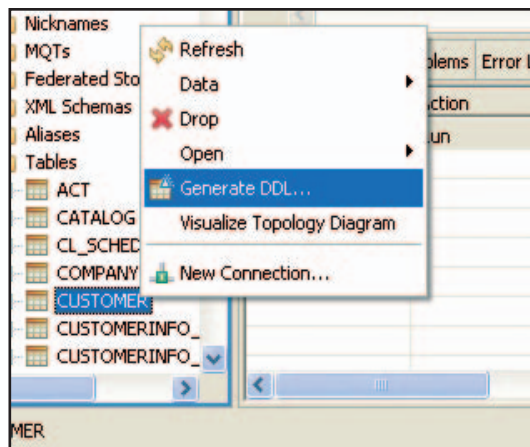



Fig. 5: ***Developer Workbench***

```
// XmlUpDel -> rollbackChanges
PreparedStatement stmt1 = con
   .prepareStatement("DROP TABLE
                              oldcustomer");
stmt1.execute();
```

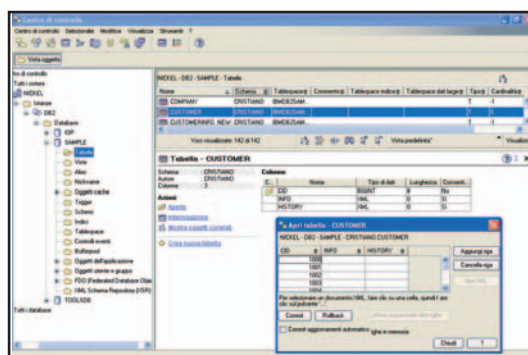



Fig. 6: ***Pannello di controllo***

Questi metodi sono generali e si usano per tutti i database e danno un quadro completo su come usare da subito DB2 Viper.

CONNESSIONE ED ERRORI

In tutti i file usati negli esempi precedenti ci si è soffermati sulle caratteristiche innovative del DB2 Viper, eppure un ruolo decisivo lo svolgono anche altre classi che consentono di connettersi al db e di gestire le eccezioni. La connessione è il primo aspetto da considerare per potersi collegare al db e in ogni classe ci sono alcune righe che la effettuano.

```
// In tutte le classi
String[] arg = { "SAMPLE", "localhost", "50000",
                              "user"  "pwd" };
Db db = new Db(arg);
Connection con;
con = db.connect();
```

Nelle varie classi si usano queste poche righe che vanno riconfigurate a seconda delle impostazioni di ognuno. I valori da passare sono, nell'ordine, il nome dell'istanza del db, il server al quale connettersi, la porta del db, e username e password. Con questi valori si può usare direttamente tutto il codice, a patto che si sia eseguita sulla propria macchina l'installazione standard del DB2 Viper, e che si imposti correttamente il proprio username e password.
Interessante è l'oggetto *Db* che permette di effettuare la connessione. Infatti, avendo il DB2 una lunga storia alle spalle, sono davvero molte le versioni e le tipologie di connessioni che si possono effettuare. Poiché si usa Java, la connessione più naturale da usare è quella che sfrutta il driver *jdbc*, e il tipo 4 è quello consigliato per le ultime architetture. Per connettersi effettivamente al db serve la url di connessione.

```
// Db -> connect
Properties props = new Properties();
url = "jdbc:db2://" + server + ":" + portNumber
                              + "/" + alias;
if (null != userId) {
   props.setProperty("user", userId);
   props.setProperty("password", password);
}
con = DriverManager.getConnection(url, props);
con.setAutoCommit(false);
```

L'*url* usa una parte fissa che dipende dallo specifico db e una variabile, come il numero della porta e l'alias, ovvero il nome dell'istanza del db. Poi, nell'oggetto *Properties*, si impostano username e password e tutto ciò serve per ottenere la connessione. L'ultima riga imposta il commit automatico a *false* e quindi, tutte le modifiche fatte sul db diventano effettive solo quando si esegue il metodo *db.disconnect*.

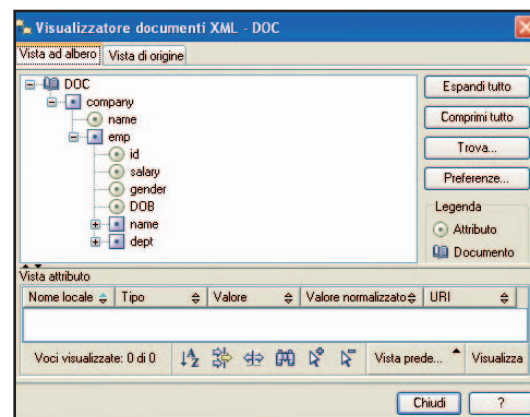



Fig. 3: ***Struttura con campo per l'indice***

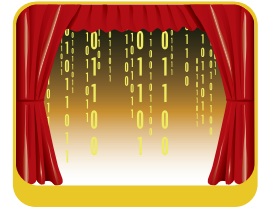
```
// Db -> disconnect
con.commit();
con.close();
```

In questo modo si possono eseguire tutte le operazioni e solo se tutto va a buon fine, allora si fa il commit. Viceversa, se si incontrano problemi, si lancia un'eccezione che fa il rollback delle operazioni. Questo è il compito della classe JdbcException.

```
// JdbcException -> handle
if (conn != null) {
  try {
    conn.rollback();
  } catch (Exception e) {}
```

Con questo codice si hanno tutte le informazioni necessarie per costruire un'applicazione funzionante che sfrutta le caratteristiche innovative del DB2 Viper.

STRUMENTI DI SVILUPPO E AMMINISTRAZIONE IBM

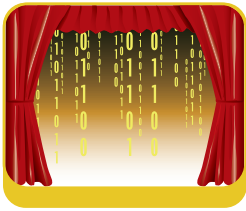
IBM mette a disposizione un pacchetto completo di sviluppo del codice e gestione del db per poter avere un ambiente integrato che consenta di lavorare più facilmente e velocemente. *Developer Workbench* serve per scrivere righe di codice Java avendo sotto mano il db. Infatti, un aspetto utile è la possibilità di avere informazioni sul db usato semplicemente aprendo la prospettiva *Data*. Inoltre, essendo basato sul progetto *Eclipse*, si ritrovano molti degli elementi di sviluppo caratteristici di questo ide. La foto mostra tutti gli aspetti che si possono sfruttare di questo ambiente.
Infatti, nella parte centrale, ci sono tutte le classi del progetto e, a destra, la visualizzazione dei metodi della classe selezionata. Mentre nel resto dello schermo ci sono le informazioni sul db. A sinistra, c'è la navigazione del db attraverso lo schema e le tabelle. In basso è messa in risalto proprio la possibilità di visualizzare i dati contenuti nel db. Inoltre, si possono cancellare drop, tabelle, oppure generarne lo script di creazione, *DDL*. Quindi, già solo con questo ambiente di sviluppo si ha sotto controllo l'intero db.
Per chi, invece, vuole una gestione più accurata c'è un apposito pannello di controllo per l'amministrazione del db.
Dal pannello si possono ottenere le stesse informazioni sulle tabelle, con la struttura usata, il nome e il tipo dei campi e i dati memorizzati. Si può anche navigare facilmente *XML Schema Repository*, in basso a sinistra, con il quale si ottiene la figura sullo schema *Customer*, usata in precedenza. Con questi strumenti si ha il controllo totale del codice sorgente e del db usato.

INDICI PER CAMPI XML

Dopo aver visto le operazioni fondamentali sul db e gli strumenti per operare al meglio, si possono considerare funzionalità avanzate che danno maggior controllo del db. Sempre per cercare di dare al formato xml gli stessi vantaggi del db relazionale, DB2 Viper consente di realizzare indici per reperire i dati in maniera veloce. Inoltre, come per il relazionale, si possono creare indici unique, per controllare che i dati inseriti in uno specifico campo siano, appunto, unici.

```
// XmlIndex -> TbIndexUniqueConstraint1
String create = "CREATE TABLE COMPANY(id INT,
                docname VARCHAR(20), doc XML)";
stmt.executeUpdate(create);
stmt = con.createStatement();
stmt.executeUpdate("CREATE UNIQUE INDEX
                empindex on company(doc)"
  + "GENERATE KEY USING XMLPATTERN
                '/company/emp/@id'"
  + "AS SQL  DOUBLE ");
```

Dopo aver generato la tabella *company*, composta dal campo *doc* di tipo xml, si crea l'indice *unique* solo su un campo specifico del campo *doc*. In altre parole, il campo *doc* contiene xml e all'interno dell' xml c'è l'elemento company formato dall'elemento *emp* che contiene a sua volta un altro elemento, *id*, sul

```
DB2 CLP - DB2
C:\Cristiano\Programmazione\DB2Viper\DB2Server91\BIN>db2 -vtf RunstatsCmd.db2
CONNECT TO SAMPLE

   Informazioni sul collegamento del database

 Server database            = DB2/NT 9.0.0
 ID autorizzazione SQL      = CRISTIAN...
 Alias database locale      = SAMPLE


RUNSTATS ON TABLE CRISTIANO.CUSTOMER
DB20000I  Il comando RUNSTATS Þ stato completato con esito positivo.

RUNSTATS ON TABLE CRISTIANO.CUSTOMER ON COLUMNS (Info, History)
DB20000I  Il comando RUNSTATS Þ stato completato con esito positivo.

RUNSTATS ON TABLE CRISTIANO.CUSTOMER ON COLUMNS(Info, History LIKE STATISTICS) W
ITH DISTRIBUTION ON KEY COLUMNS DEFAULT NUM_FREQVALUES 30 NUM_QUANTILES -1 ALLOW
 READ ACCESS
DB20000I  Il comando RUNSTATS Þ stato completato con esito positivo.

RUNSTATS ON TABLE CRISTIANO.CUSTOMER ON COLUMNS(Info, History LIKE STATISTICS) W
ITH DISTRIBUTION ON KEY COLUMNS EXCLUDING XML COLUMNS
DB20000I  Il comando RUNSTATS Þ stato completato con esito positivo.

CONNECT RESET
DB20000I  Il comando SQL Þ stato completato con esito positivo.


C:\Cristiano\Programmazione\DB2Viper\DB2Server91\BIN>_
```

Fig. 3: ***Esecuzione script per le statistiche***

quale impostare l'indice unico. La figura, catturata dal pannello di controllo, mostra tutta la struttura e il campo id per il quale c'è l'indice.
A questo punto si può passare alle prove: affinché sia un indice unico, si deve poter inserire un solo elemento con lo stesso id nel campo doc.

```
// XmlIndex -> TbIndexUniqueConstraint1

stmt.executeUpdate("INSERT INTO company
                          values (1, 'doc1', xmlparse "
   + "(document '<company
        name=\"Company1\"> <emp id=\"31201\""
   + " salary=\"60000\" gender=\"Female\"
                              DOB=\"10-10-80\">"
...
stmt.executeUpdate("INSERT INTO company
                          values (1, 'doc1', xmlparse "
   + "(document '<company
        name=\"Company2\"> <emp id=\"31201\""
   + " salary=\"50000\" gender=\"Male\"
...
```

Il codice sopra inserisce correttamente la prima riga, ma non la seconda, per la quale si genera un'eccezione. La causa è l'indice unico sul campo *id* che non autorizza a memorizzare due righe che abbiano lo stesso valore, 31201 in questo caso. Inoltre, l'indice creato si aspetta valori di tipo *SQL DOUBLE*, cioè numeri, e quindi, tentando di inserire id formato da stringhe, si genera nuovamente un errore. Se invece si vuole creare un indice che accetti stringhe, si può utilizzare un altro metodo.

```
// XmlIndex -> TbIndexVarcharConstraint
stmt.executeUpdate("CREATE INDEX empindex on
                                company(doc)"
   + "GENERATE KEY USING XMLPATTERN
                             '/company/emp/@id'"
   + "AS SQL VARCHAR(4)");
```

La caratteristica di questo nuovo indice è di permettere id in formato stringa e al massimo di 4 caratteri.

STATISTICHE PER TABELLE E INDICI

Dopo aver creato le tabelle, averle popolate e aver reso le query più veloci con gli indici, si deve tenere il tutto aggiornato. Le query, infatti rendono al meglio sfruttando gli indici più opportuni, solo se le statistiche su tabelle e campi sono attuali. Per far ciò, si può usare la classe *XmlRunstats*.

```
// XmlRunstats -> xmlRunstats

File outputFile = new File("RunstatsCmd.db2");
FileWriter out = new FileWriter(outputFile);
String cmd1 = "RUNSTATS ON TABLE " +
                                  fullTableName;
String cmd2 = "RUNSTATS ON TABLE " +
                                  fullTableName
   + " ON COLUMNS (Info, History) ";
String cmd3 = "RUNSTATS ON TABLE " +
                                  fullTableName
   + " ON COLUMNS(Info, History LIKE
                                  STATISTICS)"
   + " WITH DISTRIBUTION ON KEY COLUMNS"
   + " DEFAULT NUM_FREQVALUES 30
                              NUM_QUANTILES -1"
   + " ALLOW READ ACCESS";
String cmd4 = "RUNSTATS ON TABLE " +
                                  fullTableName
   + " ON COLUMNS(Info, History LIKE
                                  STATISTICS)"
   + " WITH DISTRIBUTION ON KEY COLUMNS"
   + " EXCLUDING XML COLUMNS ";
Process p = Runtime.getRuntime().exec("db2 -vtf
                             RunstatsCmd.db2");
```

Per lanciare le statistiche, si crea un file *RunstatsCmd.db2*, lo si popola delle istruzioni da compiere, e lo si lancia. Nel metodo, le istruzioni servono per eseguire le statistiche sulla tabella, sulle colonne, su tutte le partizioni, consentendo l'accesso in lettura e scrittura durante la generazione, oppure escludendo alcuni campi per avere maggior velocità nella generazione.
Poiché si genera un file, un'altra possibilità è di eseguirlo dalla finestra dei comandi. Mettendosi nella directory *BIN* relativa a dove si è installato il DB2 Viper, si può lanciare la stessa istruzione eseguita via codice, come testimoniato dalla figura.

REGISTRAZIONE SCHEMA XML

Negli esempi precedenti si è capito l'importanza che rivestono gli schema per validare la struttura xml e si è visto come il pannello di controllo possa visualizzarli. Manca come registrarli nel db. E questo è il compito della classe *XmlSchema*. Lo schema da registrare è relativo agli ordini e ha la sua struttura nel file *order.xsd*. Guardano il file, si può notare come, oltre alla definizione di campi specifici dell'ordine, ci sono anche delle inclusioni di altri

file, come *customer.xsd*. In questo modo si divide il lavoro e si impone che i dati da inserire debbano seguire la struttura contenuta in *order* e anche quella contenuta in *customer*.

```
// XmlSchema -> registerXmlSchema

CallableStatement callStmt = con
   .prepareCall("CALL
         SYSPROC.XSR_REGISTER(?,?,?,?,NULL)");
File xsdFile = new File("order.xsd");
FileInputStream xsdData = new
                         FileInputStream(xsdFile);
...
callStmt.setBinaryStream(4, xsdData, (int)
                               xsdFile.length());
callStmt.execute();
addXmlSchemaDoc(con, "customer.xsd");
```

Con la chiamata al processo si registra un nuovo schema. Uno dei valori che si passa è proprio il file *order.xsd*. Poiché, come detto, serve registrare anche *customer.xsd*, si deve aggiungere allo schema anche il nuovo file. Il metodo per aggiungere un altro file xml allo schema precedente è simile e cambia solo nella procedura DB2 chiamata.

```
// XmlSchema -> addXmlSchemaDoc

CallableStatement callStmt = con
   .prepareCall("CALL
  SYSPROC.XSR_ADDSCHEMADOC(?,?,?,?,NULL)");
```

A questo punto si può verificare la registrazione dello schema chiamando l'ultimo metodo, *insertValidatexml*, che compie operazioni simili a quanto visto negli esempi precedenti.

XML DAL DB

Negli esempi precedenti si vede come eseguire istruzioni da java con query sql. Volendo, si possono eseguire le stesse operazioni, direttamente dal db, usando le *stored procedure*, procedure memorizzate nel db. Il tutto poi può essere lanciato da Java. Molti elementi in gioco, ma i passi da compiere sono abbastanza facili.

```
// RelToXmlDoc -> callRelToXmlProc
String procName = "RELTOXMLPROC";
String sql = "CALL " + procName + "()";
CallableStatement callStmt =
                          con.prepareCall(sql);
callStmt.execute();
ResultSet rs = callStmt.getResultSet();
fetchAll(rs);
```

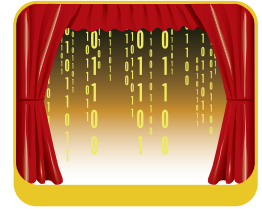
Con queste poche istruzioni java si lancia una procedura memorizzata nel db. Una volta recuperato il *ResultSet* ottenuto dalla procedura, il metodo *fetchAll* si occupa di formattare il risultato in maniera opportuna. In realtà, quindi, il grosso del lavoro viene eseguito nel db, dalla procedura. A parte le operazioni specifiche per creare una procedura e dichiarare un cursore, cioè un oggetto dove memorizzare il risultato della query, le istruzioni sono quelle viste negli esempi precedenti usando il codice java.

```
// PROCEDURE reltoxmlproc()
SELECT po.PoNum, po.CustID, po.OrderDate,
...
XMLELEMENT (NAME "PurchaseOrder"
...),
XMLATTRIBUTES (po.CustID as "CustID",
         po.PoNum as "PoNum", po.OrderDate as
            "OrderDate", po.Status as "Status" ),
  XMLELEMENT (NAME "CustomerAddress",
                                  XMLCONCAT(
  XMLELEMENT (NAME "e.Name", c.Name ),
...
```

Quindi, le istruzioni *SQL/XML* viste nei vari esempi, si possono usare in modo simile anche nelle *stored procedure*.

CONCLUSIONI

IBM, dopo aver segnato la storia dei database, tenta di sconvolgere nuovamente le regole, mettendo a segno il primo database ibrido: relazionale e xml. Per poterlo sfruttare al meglio, si sono visti esempi per poter recuperare dati, modificarli, creare nuove tabelle, indici su campi xml e anche chiamare *stored procedure* direttamente nel db. Le classi create sono davvero semplici e offrono una base concreta dalla quale partire per personalizzare la propria applicazione potendo eseguire moltissime funzionalità messe a disposizione da DB2 Viper. Il db ibrido e gratis.

Cristiano Bellucci

SVILUPPO E SERVER

Oltre a supportare lo sviluppo in java, il DB2 Viper è integrato in VS.NET e in MS Visual Studio 2005. Lato server ci sono versioni disponibili per Windows, Linux, Solaris, AIX e HP sia a 32 che a 64 bit.

LA SICUREZZA CON IL NUOVO IIS 7.0

INTERNET INFORMATION SERVICES PRESENTA DIVERSE NOVITÀ ARCHITETTURALI CHE LO RENDONO ANCORA PIÙ AFFIDABILE E DIFFICILMENTE ATTACCABILE. GRAZIE ALL'AGGIUNTA DEL SUPPORTO DIRETTO AL .NET FRAMEWORK E LA FORTE COMPONENTIZZAZIONE.

La sicurezza di un web server costituisce una buona parte della sicurezza di un'applicazione . Poggiare su fondamenta solide aiuta infatti a beneficiare della protezione intrinseca che è offerta dal web server, evitando di soffermarsi troppo sulle funzionalità di sicurezza di base, che sono garantite a livello centralizzato. Nel caso di IIS, le versioni dalla 5.0 in poi hanno sempre cercato di garantire allo sviluppatore maggiori funzionalità di sicurezza, così da rendere possibile una convivenza più semplice anche con l'altra metà del cielo, cioè i sistemisti. IIS 7.0, d'altra parte, garantisce le stesse funzionalità già presenti in IIS 6.0, con in più un insieme di nuove funzionalità che lo rendono ancora più interessante.

L'EVOLUZIONE DEL CONCETTO DI SICUREZZA

IIS 6.0 ha rappresentato sul versante della sicurezza complessiva del server web un passo in avanti molto importante, con l'introduzione dei meccanismi di protezione propri di URLScan. **URLScan** è un tool aggiuntivo, che può essere installato su IIS 5.0 o 6.0 (su quest'ultimo aggiunge soltanto un numero limitato di funzionalità) e che consente di filtrare, come il nome stesso suggerisce, le richieste in base al contenuto dell'URL. Questo genere di filtri è molto comodo, oltre che vitale, per evitare attacchi che sfruttino l'**URL canonicalization**, un attacco particolare che mira ad inviare richieste che il server web può fraintendere, per servire risorse che altrimenti non sarebbero state raggiungibili.
Il tipico caso è quello del carattere \ che se inserito all'intero di una sequenza ben definita, ad esempio "..\", può assumere un significato totalmente differente.
Supponendo di richiamare qualcosa come http://localhost/..\..\..\windows/system32\cmd.exe, si potrebbe arrivare tranquillamente a cmd.exe, che come si sa corrisponde al prompt dei comandi di Windows. Da lì a far eseguire applicazioni o comandi il passo è davvero breve.
Anche se oggi è difficile trovare un server web tra quelli più diffusi che soffra di problemi del genere, in passato la probabilità di una vulnerabilità era più che elevata. Ovviamente al giorno d'oggi i problemi di sicurezza sono più sentiti e le tecniche di attacco più raffinate, a tal punto che IIS 7.0, la prossima versione di IIS, dedica molte risorse alla risoluzione di questo problema.
La prima prerogativa di questa nuova versione è la forte **componentizzazione**, che porta vantaggi sia dal punto di vista della configurazione, garantendo la possibilità di configurare a proprio piacere l'applicazione, sia dal punto di vista della sicurezza. La riduzione della superficie d'impatto, infatti, è garanzia di una minore esposizione esterna, poiché meno funzionalità disponibili ci sono, minore è il rischio che siano malconfigurate e lascino aperta un'eventuale porta di entrata.
In questo senso, sin da IIS 6.0 sono state lasciate attive solo le funzionalità minime, ma con IIS 7.0, grazie all'uso dell'**applicationHosting.config**, l'intera configurazione è possibile semplicemente editando un file XML, piuttosto che utilizzando un tool visuale o da riga di comando. La rivoluzione, semplicemente, sta nella possibilità di configurare qualsiasi funzionalità da questo file. Ed a pieno titolo tra ciò che si può modificare rientrano anche le impostazioni di sicurezza.
È così ad esempio possibile specificare, anche a livello di singolo sito, i modules o handlers da utilizzare, rendendo possibile la creazione anche di un semplice sito che sia in grado di supportare solo file statici, piuttosto che applicazioni ASP o ASP.NET, in maniera indipendente da quello che poi gli altri siti o applicazioni dello stesso server potranno fare. Per evitare che il browsing delle directory che non

hanno una pagina di default sia attivo, ad esempio, è sufficiente inserire nel web.config locale al sito, oppure all'interno **dell'applicationHosting.config**, una sezione di configurazione simile alla seguente:

```
<configuration>
  <system.webServer>
    <directoryBrowse enabled="false" />
  </system.webServer>
</configuration>
```

Questo approccio dichiarativo alla configurazione è peraltro già noto agli sviluppatori ASP.NET, dato che il web.config è addirittura condiviso, senza necessità di avere due file separati per la configurazione e con il vantaggio, tutt'altro che relativo, di poter controllare tutta la configurazione da un unico punto.

UN URLSCAN INTEGRATO NEL MOTORE

Questo livello di granularità consente di proteggere in maniera adeguata le applicazioni, anche grazie alla possibilità di inserire nei file di configurazione, policy di tipo più avanzato, che ricalcano per certi versi quelle offerte da URLScan.
È ad esempio possibile specificare IP che non possono avere accesso, piuttosto che negare particolari sequenze contenute nell'URL, a garanzia che non possano essere sfruttati dall'esterno alcuni tra gli attacchi più diffusi.
Molto interessante il meccanismo che equivale al filtro sulle estensioni attivato con URLScan, che **consente di negare l'accesso a particolari estensioni di file**. Questa tecnica è spesso utilizzata, con stratagemmi diversi, per evitare che gli utenti possano mettere online file con estensioni particolari e farli scaricare. Tra l'altro, è interessante notare come questo meccanismo aiuti anche la sicurezza passiva in un caso ben particolare, quando ad esempio il supporto per ASP.NET dovesse essere disabilitato. In questa circostanza, le relative estensioni non sarebbero mappate sotto IIS, con il risultato che una richiesta in realtà produce semplicemente il sorgente della risorsa, ad esempio della pagina. Sfruttando questo meccanismo, invece, le estensioni più comuni sono protette a prescindere dall'attivazione del corrispondente motore. Segue un esempio di quello che contiene un tipico applicationHosting.config, che nega la visualizzazione di file .asa e .master, rispettivamente necessari per il global.asa di ASP o le Master Page di ASP.NET 2.0.

```
<configuration>
  <system.webServer>
    <requestFiltering>
      <fileExtensions allowUnlisted="true">
        <add fileExtension=".asa" allowed="false" />
        <add fileExtension=".master" allowed="false"
                                                  />
      </fileExtensions>
  </system.webServer>
</configuration>
```

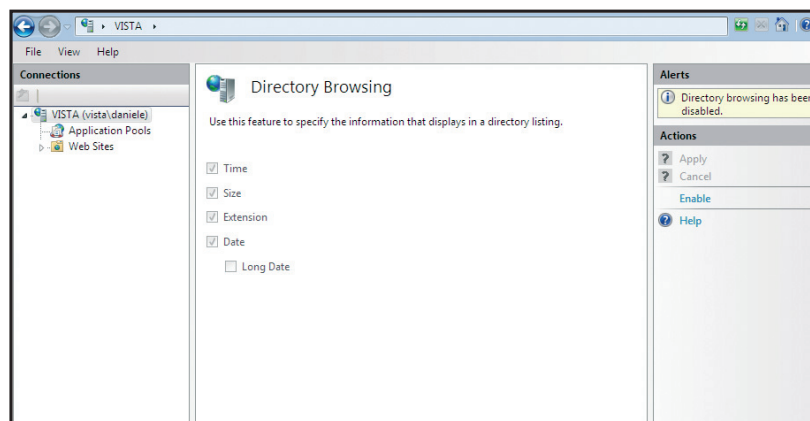



Fig. 1: ***L'interfaccia di configurazione del directory browsing integrato in IIS 7.0.***

Oltre che sull'estensione, è possibile agire anche sul VERB HTTP utilizzato, negando, ad esempio, quello PUT, qualora non si utilizzi WebDav.
Particolarmente interessante è invece la sezione **hiddenSegments**, perché nasconde del tutto le risorse specificate, generando un errore 404. Questa sezione è complementare a quella delle estensioni, poiché consente invece di agire sul

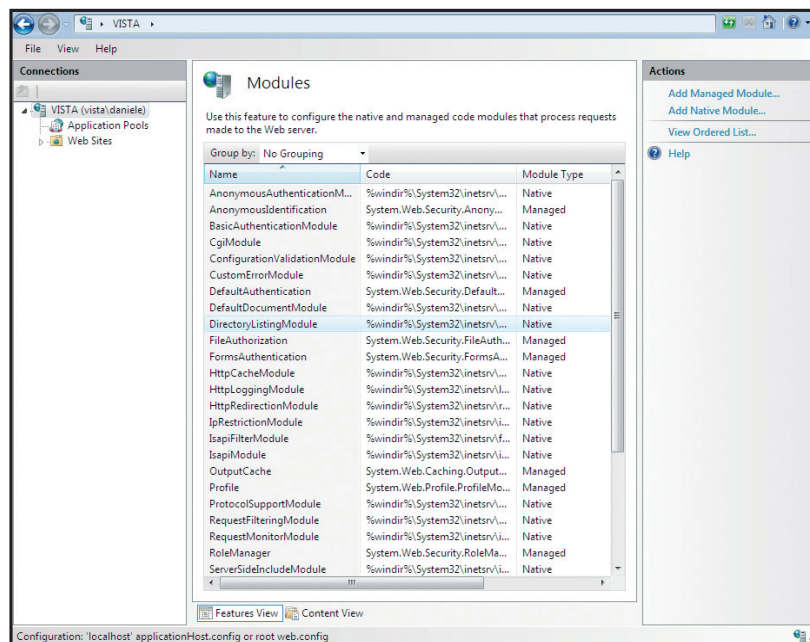



Fig. 2: ***Tutti i moduli, managed e nativi, installati di default in IIS 7.0.***

resto della stringa corrispondente all'URL, facendo risultare come non esistenti alcuni percorsi dedicati a file di sistema.

```
<configuration>
  <system.webServer>
    <requestFiltering>
      <hiddenSegments>
        <add segment="web.config" />
        <add segment="App_Data" />
      </hiddenSegments>
  </system.webServer>
</configuration>
```

Nell'esempio si può notare come vengano protetti il file web.config e la cartella /App_Data/, utilizzati il primo da ASP.NET ed IIS 7.0 per la configurazione, il secondo per il salvataggio di file applicativi (database Access, file XML o simili) di cui non si vuole consentire il download. Infine, molto importante la caratteristica di poter specificare all'interno di questo file, che è un documento XML, quindi manipolabile da un parser, anche gli IP a cui negare l'accesso. In questo caso la sezione è *ipSecurity* e la sintassi è abbastanza chiara. Nell'esempio viene dato accesso a tutti gli IP, tranne quelli specificati.

```
<system.webServer>
  <security>
    <ipSecurity allowUnlisted="true">
      <add ipAddress="192.168.10.10"
                              allowed="false" />
      <add ipAddress="192.168.5.101"
                              allowed="false" />
    </ipSecurity>
  </security>
</system.webServer>
```

La possibilità di copiare e sincronizzare questa sezione attraverso script apre scenari importanti, poichè consente di bloccare eventuali IP da cui provengono attacchi in maniera molto più semplice.

LE VERSIONI DI IIS 7.0 SU WINDOWS VISTA

IIS 7.0 sarà disponibile in 3 versioni, a seconda del sistema operativo su cui andrà a girare. Come in passato, non sarà possibile fare l'upgrade di IIS senza aggiornare il sistema operativo. A differenza del passato, anche sulle versioni non server sarà possibile creare quanti siti web si vuole.
Ne saranno prive le versioni Starter e Home Basic, mentre con la versione Home Premium si avrà la limitazione di 3 connessioni contemporanee e con le versioni Business, Enterprise ed Ultimate il limite sarà portato al classico 10, già disponibile oggi per Windows XP. In ogni caso, non ci saranno differenze con la versione che sarà distribuita insieme a Windows Longhorn Server.

ESTENDERE LA SICUREZZA CON GLI HTTPMODULE

Dal punto di vista dell'estendibilità, gli sviluppatori ASP.NET già conoscono gli **Http Module**, il sistema attraverso il quale si può estendere la pipeline di richiesta e risposta praticamente all'infinito, aggiungendo o modificando le funzionalità di base offerte dall'engine.
IIS 7.0 introduce la novità, molto importante, di offrire questa pipeline anche sulle richieste che non siano gestite da ASP.NET. Per fare questo, esistono due tipi di application pool, il classico di IIS 6.0, che usa solo moduli unmanaged, e l'**integrated**, che invece sfrutta gli HttpModule managed anche per gli altri tipi di risorse, integrando direttamente il supporto per il .NET Framework.
Questa caratteristica consente di proteggere in maniera molto semplice le applicazioni, sfruttando le **Membership** API di ASP.NET 2.0. Una volta impostata, è poi possibile proteggere tutte le risorse di un'applicazione all'interno di un application pool di tipo integrated sfruttandolo anche per risorse come PDF, Word o pagine ASP, piuttosto che PHP.
Il tutto è possibile perché l'evento **Authorize Request**, che una volta era intercettabile solo per richieste a risorse ASP.NET, in questa particolare modalità ora è scatenato direttamente da IIS, così che nella pratica possa essere sfruttato da qualsiasi tipo di risorsa.
Da questo punto di vista, la creazione di un sistema di protezione personalizzato è ancora più semplice, dato che il sistema di autorizzazione è facilmente estendibile proprio sfruttando gli HttpModule.
ASP.NET ha introdotto il concetto di HttpModule sin dalla sua prima versione, dunque ciò che offre l'integrazione con IIS 7.0 non è esattamente rivoluzionario in senso assoluto, quanto per la portata di queste funzionalità, che come detto ora non sono più limitate alle sole risorse managed.
È così possibile estendere facilmente alcuni comportamenti di IIS, come la pagina offerta per la visualizzazione dei file presenti in una directory che non abbia pagina di default, piuttosto che l'aggiunta di un watermark sulle immagini.
È necessario dunque creare un HttpModule, aggiungendo una classe che implementi l'interfaccia **IHttpModule**. Per farlo, è sufficiente

utilizzare Visual Studio 2005, anche Express (nelle varianti VB o C#, a seconda del linguaggio scelto, perché Visual Web Developer non permette la creazioni di class library).
Un HttpModule si contraddistingue per il fatto che ha un metodo **Init**, nel quale vanno registrati gli event handler associati agli eventi dell'applicazione, ed uno Dispose, all'interno del quale vanno eventualmente scaricate le risorse unmanaged. Nel nostro caso, per creare una nuova pagina per il directory browsing che offra maggior sicurezza, è sufficiente registrarsi per l'evento **BeginRequest**, che come il nome suggerisce si verifica ad ogni richiesta di pagina, non appena questa comincia. Nell'event handler associato, andiamo poi a verificare che non esista su disco una pagina di default e nel caso, andiamo a scrivere sullo stream di risposta l'elenco dei file, con un po' di grafica associata. Il tutto è abbastanza semplice, fa uso di un po' di classi del namespace System.IO, ma può dare un risultato davvero interessante, oltre che estremamente personalizzato, come si può notare nella figura qui mostrata. In questo caso, ad esempio, è implementato un semplice controllo che, sfruttando le Roles API di ASP.NET 2.0, mostra la pagina con l'elenco dei file solo agli utenti che appartengono al ruolo administrator, mantenendo la sicurezza sempre alta, perché l'utente normale non vedrà in dettaglio il contenuto della directory, mentre un amministratore potrà visionarla in tutta sicurezza. Eventualmente questa tecnica può essere applicata con successo e relativa facilità anche in altri ambiti (ad esempio per gli errori di tipo 500), per aumentare la sicurezza di tutte le funzionalità del web server, da un unico punto centralizzato.

Fig. 3: *Il nostro DirectoryListingModule personalizzato in azione.*

```
<configuration>
  <system.webServer>
    <anonymousAuthentication enabled="true"
            userName="" defaultLogonDomain="" />
  </system.webServer>
</configuration>
```

Ovviamente, così come nel caso di IIS 6.0, è possibile modificare queste impostazioni anche sulla base di ogni singolo application pool. Il risultato è che diventa possibile sfruttare la sicurezza derivante dall'uso di questo utente con privilegi minori, per far sì che eventuali attacchi ad un sito web non possano lanciare processi esterni, piuttosto che leggere nel registry o creare file in directory non consentite.

NUOVO MODELLO DI SICUREZZA DEI PROCESSI

Da IIS 3.0 siamo stati abituati a convivere con l'utente **IUSR_nomecomputer** e con i problemi derivanti da un'errata configurazione dei permessi.
In IIS 7.0 questo utente è stato rimosso, così come il gruppo **WGP_IIS**, per far posto all'utente **IUSR**, che è disponibile direttamente all'interno del sistema operativo. Si tratta di un nuovo account con privilegi minimi, che è in grado di fare ben poche cose.
Di default IIS 7.0 utilizza l'autenticazione anonima del processo, gira cioè con l'account specificato all'interno dell'application Hosting. config.

CONCLUSIONI

La sicurezza di IIS 6.0 è già a prova di bomba: lo garantisce il fatto che non ci sono state, praticamente, vulnerabilità di sicurezza note in questi anni. Il lavoro in tal senso, per quanto concerne IIS 7.0, è più che altro rivolto ad un consolidamento di questa caratteristica, grazie all'uso di una forte componentizzazione ed all'integrazione con il CLR del .NET Framework, che consente di plasmare a proprio piacere le funzionalità esposte, arrivando a poter modificare comportamenti del web server che in passato non era possibile toccare. Non ci resta che attendere l'uscita definitiva di Windows Vista per poter utilizzare tutte queste funzionalità, anche se già ora, nelle CTP e nella beta 2, è possibile toccarlo con mano.

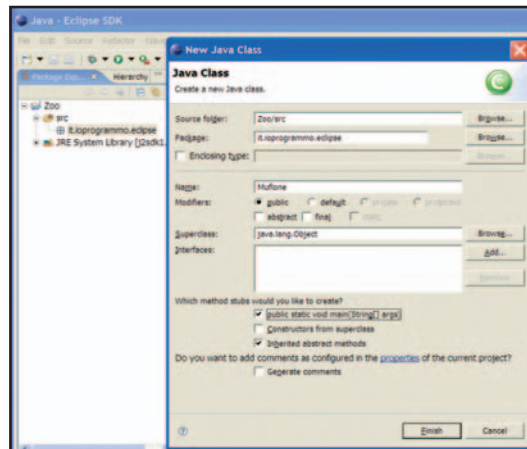
Daniele Bochicchio

Daniele Bochicchio è il content manager di ASPItalia.com, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

JAVA ED ECLIPSE IN SETTE PASSI

È POSSIBILE IMPARARE AD USARE ECLIPSE, IL MIGLIOR SISTEMA DI SVILUPPO JAVA, IN POCO PIÙ DI UN'ORA? NON PROPRIO. PERÒ SI POSSONO INIZIARE A FARE I PRIMI PASSI. LA PRIMA APPLICAZIONE COMPLETA CI ATTENDE...

Prima buona notizia: non dovete installare niente. Se il vostro sistema ha già un runtime di Java, o meglio ancora un Java SDK, vi basta scompattare il file ZIP di Eclipse nella root del disco, o da qualsiasi altra parte. Il risultato è una directory *eclipse*, che contiene un eseguibile eclipse.exe (per la versione Windows). Lanciatelo. Eclipse chiederà un *workspace*, la directory che contiene le impostazioni e il codice dei progetti. Qualsiasi directory vuota va bene. Se non esiste, Eclipse provvederà a crearla. Potete comunque cambiare workspace in qualsiasi momento, e anche averne più di uno.
Subito dopo Eclipse presenta una schermata di benvenuto. Basta chiuderla, con un click sulla crocetta nel tab in alto a sinistra, per trovarsi di fronte all'IDE. All'inizio, l'unica prospettiva visibile è la *Java Perspective* (il box *Dimmi quattro parole* spiega gli elementi del workspace, prospettive comprese).
Proviamo subito a creare un progetto: menu *File->New->Project*. Si aprirà un wizard. Selezionate Java Project e premete su Next. Date un nome al progetto (Zoo) e scegliete Create separate source and output folders sotto Project Layout.

Così Eclipse userà due directory separate per il sorgente (cartella src) e per le classi compilate (cartella bin). Lasciate perdere le altre opzioni e cliccate su Finish. Vedrete il nuovo progetto nel Package Explorer, sulla sinistra dell'IDE. Per ora contiene solo la directory src, ancora vuota, e le librerie di Java. (Normalmente, Eclipse trova da sé la JVM del sistema e l'aggiunge al progetto. Se questo non dovesse succedere, se avete più di un runtime Java e volete scegliere quale usare, andate nel menu Window->Preferences, selezionate Java e poi Installed JREs. Da qui potrete cercare tutti i runtime di Java e selezionare quello preferito da usare di default).
Per creare un package, aprite il menu contestuale della directory src e scegliete New->Package . Date un nome al package: *it.ioprogrammo.eclipse*. Ora potete aprire il menu contestuale del package per crearci dentro una classe: New->Class. Nella finestra New Java class,

chiamate la classe Muflone e controllate che il package sia it.ioprogrammo.eclipse. Selezionate anche l'opzione public static void main... per dire a Eclipse che la classe deve contenere un main() vuoto. Premete Finish.
Ora dovreste vedere la classe Muflone nel Package Explorer sotto it.ioprogrammo.eclipse, e il suo codice nell'editor.
Guardate anche la vista Outline sulla destra: mostra tutti i metodi e i campi della classe selezionata. Per ora l'unico metodo è il main(), indicato con un pallino verde (perché è public) e una piccola "s" (perché è static). L'Outline diventa utile quando i metodi sono tanti, anche perché permette di filtrarli e ordinarli in base al nome e alla visibilità.

REQUISITI

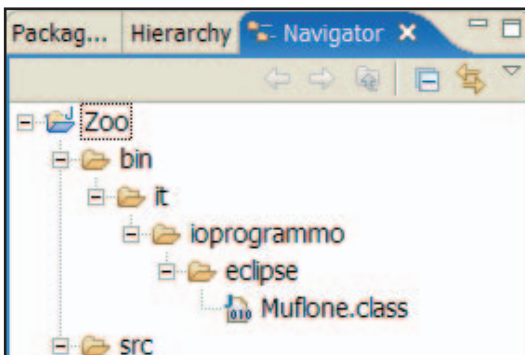
Conoscenze richieste
Basi di Java

Software
Una qualsiasi versione del runtime o dell'SDK di Java, Eclipse 3.2.

Impegno

Tempo di realizzazione

NAVIGARE I FILE

Esistono diverse configurazioni della finestra principale di Eclipse, chiamate prospettive (vedi box Dimmi quattro parole). La prospettiva di default è la Java Perspective. Le altre appaiono magicamente quando servono, oppure potete aprirle dal menu Window -> Open Perspective. Ad esempio c'è una prospettiva Debug, una prospettiva Resources per gestire i file delle risorse, eccetera. Per ora restiamo nella prospettiva Java. Il Package Explorer mostra i progetti e la loro struttura, non i file sul disco. Per vedere i file serve un'altra vista che si chiama Navigator. Questa vista è disponibile di default nella prospettiva Resources, ma a me piace aggiungerla anche alla prospettiva Java. Proviamo ad aprirla, selezionando Window->Show View->Navigator dal menu principale. Il Navigator appare nella stessa posizione del Package Explorer, e si può passare dall'uno all'altro usando le tab.

Se la posizione non vi piace potete spostarlo in giro, ma non fatevi troppi problemi: di solito il Navigator si usa in alternativa al Package Explorer, quindi non c'è un gran bisogno di averli insieme sullo schermo. Il Navigator, al contrario del Package Explorer, non "capisce" i progetti, ma solo il file system. Non contiene package e classi, ma directory e file. In pratica è una versione integrata dell'Explorer di Windows che permette di fare operazioni sul file system (Rename, Delete, eccetera) senza uscire da Eclipse. Nel nostro progetto il Navigator mostra la directory src, che contiene i sorgenti, ma anche la directory bin (invisibile sotto il Package Explorer) che contiene i file compilati. Invece non si vedono le librerie di Java, che fanno parte del progetto da un punto di vista logico ma non sono contenute nella directory. I file che cominciano con un punto, come .classpath e .project, sono file XML di configurazione. Eclipse li gestisce da sé, ma se vi sentite avventurosi o curiosi potete aprirli ed editarli a mano.

LANCIARE UN PROGRAMMA

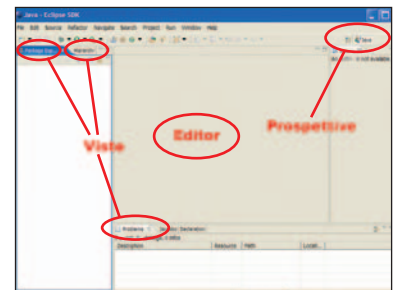
Ora che abbiamo una classe con un main(), possiamo scrivere il nostro programma. Se non avete già il sorgente di Muflone nell'editor, apritelo con un doppio click sulla classe nel Package Explorer. Nel main(), sostituite il commento piazzato da Eclipse con una versione del solito "Salve, Mondo" più orientata ai mufloni:

```
public class Muflone {
  public static void main(String[] args) {
    System.out.println("Muuuu!");
  }
}
```

DIMMI QUATTRO PAROLE

Workspace – Una directory sul disco rigido che contiene sia il codice dei progetti sia i file di configurazione. Dal punto di vista della GUI, un workspace occupa l'intera finestra principale di Eclipse. Per aprire un nuovo workspace si deve chiudere il workspace precedente, o lanciare un'altra istanza di Eclipse. Ciascun workspace contiene diverse prospettive.

Prospettiva – Una configurazione della finestra principale di Eclipse mirata ad una particolare fase dello sviluppo, come la scrittura di nuovo codice, il debugging, o l'interazione con il sistema di versionamento. Si può cambiare rapidamente prospettiva cliccando uno dei tasti in alto a destra. Una prospettiva contiene editors e viste.

Vista – Un'area dell'interfaccia dedicata ad un compito specifico. L'output del programma, il browser del disco, quello delle classi e i messaggi del compilatore sono tutti viste che possono essere spostate o impilate con il drag&drop. Ciascuna vista ha la sua piccola toolbar e i suoi menu.

Editor – Simile ad una vista, ma orientato all'inserimento di informazioni. Si possono aprire più editor dello stesso tipo contemporaneamente sullo schermo, ad esempio per editare diversi file. Eclipse cerca sempre di aprire l'editor adatto a ciascun tipo di file, e può essere facilmente esteso con nuovi editor (vedi box Un IDE fatto a pezzi).

SOFTWARE SUL CD

Borland Turbo C#

IL COMPILATORE CHE TORNA!

Borland ha segnato profondamente nel corso del tempo l'evolversi della programmazione. Già agli inizi degli anni 90, quando ancora buona parte del mondo programmava in modo procedurale, Borland era arrivata sul mercato con l'Object Turbo Pascal, primo esempio di linguaggio interamente ad oggetti dedicato alla produttività individuale. Immediatamente dopo aveva fatto segnare l'ennesima rivoluzione del mercato della programmazione lanciando la moda degli IDE Rad, era la volta di Delphi. Nel corso del tempo, tuttavia aveva cambiato radicalmente strategia e si era dedicata ai prodotti ALM tralasciando quello che era il suo mercato storico di riferimento, ovvero quello degli IDE e compilatori dedicati alla produttività individuale. Oggi questa situazione è radicalmente cambiata. Borland ha appena fondato una società apposita per riprendere lo sviluppo dei suoi IDE, segno evidente di una volontà di reinvestire in questo settore.
Directory: /Borland

APACHE 2.2.3

IL WEB SERVER CHE HA FATTO INTERNET

Una colonna portante della rete. Non si contano i moduli disponibili per estenderne le funzionalità. In questo numero di ioProgrammo lo utilizziamo in congiunzione a SVN,
Directory:/ Apache

DOTNETNUKE 4.3.5 STARTER KIT

IL PORTALE "PRECOTTO" TARGATO .NET

Un CMS completo e dotato di funzionalità realmente avanzate. In questo numero presentiamo uno "Starter Kit" ovvero un template per l'ambiente Visual Studio, che consente di installare il prodotto ma anche di estenderlo
Directory:/Apache

OPENLASZLO 3.3.3

QUASI COME FLEX MA OPENSOURCE

L'idea è semplice. C'è un file XML, l'utente richiama questo file per mezzo del browser. Il compilatore sul server lo compila on the fly e restituisce all'utente una pagina flash. La tecnica è interessante perché consente di creare applicazioni WEB 2.0 senza utilizzare AJAX
Directory OpenLaszlo

APACHE TOMCAT 5.5.17

PROGRAMMA IN JAVA PER IL WEB

Chi vuole programmare in JSP non può esimersi dall'utilizzare questo Application Server. Leggero ma completo si tratta di uno dei server più diffusi per JSP. In questo numero di ioProgrammo lo utilizziamo in congiunzione ad OpenLaszlo
Directory:/ Tomcat

ECLIPSE SDK 3.2

L'AMBIENTE TUTTO FARE PER JAVA

Nato con il supporto di IBM ma completamente OpenSource, nel tempo ha visto la partecipazione di quasi tutti i colossi dello sviluppo. Di default viene proposto come già configurato per essere utilizzato con Java, tuttavia a mezzo di plugin si può estendere per quasi ogni linguaggio. **Directory:/ Eclipse**

SVN 1.4.0

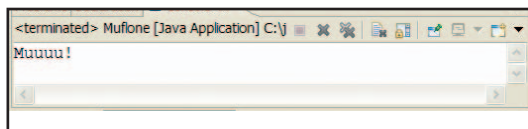
PER TENERE SOTTO CONTROLLO LE MODIFICHE

Quante volte avete eseguito una modifica al vostro codice e poi avete desiderato tornare indietro? Ecco che in questi casi un software per il controllo di versione è indispensabile. Consente di tenere traccia con dei log di tutte le modifiche apportate al vostro progetto
Directory:/ SVN

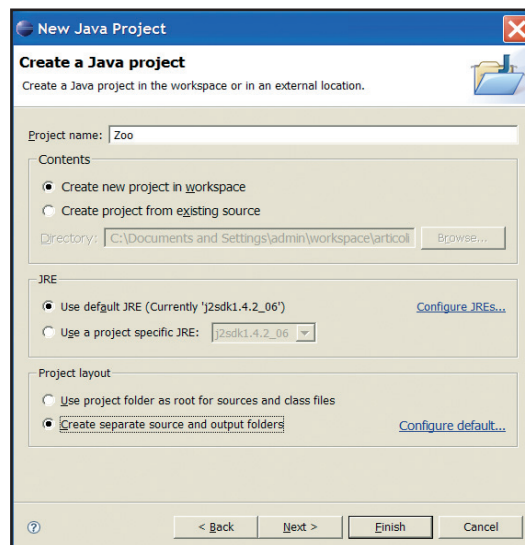
JDK 1.5.0 UPDATE 8

IL COMPILATORE PER JAVA

Se siete dei programmatori Java non potete esimervi dal conoscere questo strumento. Contiene sia il compilatore ma anche tutte le classi necessarie e gli strumenti per programmare.
Directory:/ jdk-1_5_0_08-windows-i586-p.exe

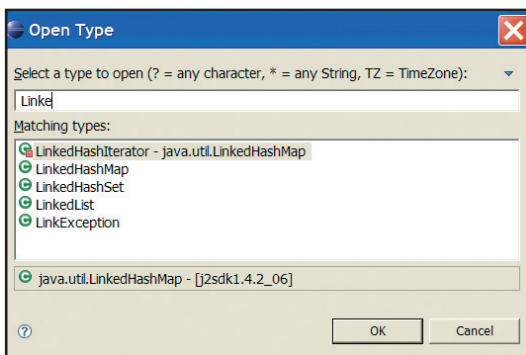
Poi salvate il file con il tasto che raffigura un floppy disk sulla toolbar di Eclipse (mi chiedo per quanto tempo ancora l'icona standard per i salvataggi resterà un floppy – ormai è un po' che non ne vedo uno vero). Potete anche usare il classico shortcut Ctrl+S sulla tastiera.
Ora dobbiamo compilare il programma, vero? No. La filosofia di Eclipse è quella di risparmiarci le operazioni ripetitive, quindi la compilazione viene eseguita automaticamente ogni volta che salviamo un file (se la cosa dovesse darci fastidio, per esempio se stiamo lavorando su un progetto molto grande, possiamo sempre disabilitare questa opzione selezionando Project->Build automatically). Quindi possiamo far finta di avere un linguaggio interpretato, e lanciare direttamente il programma. Visto che in Java ogni classe che contiene un main() è un programma, possiamo selezionare Muflone nell'editor o nel Package Explorer e scegliere Run as->Java Application dal menu contestuale.
L'output del programma apparirà sotto l'editor, nella vista Console.

Se vogliamo lanciare il programma una seconda volta, non abbiamo bisogno di passare di nuovo attraverso i menu. Sulla toolbar c'è un comodo tasto verde con una freccetta che lancia l'ultimo programma lanciato in precedenza. Oppure possiamo usare lo shortcut da tastiera Ctrl+F11.

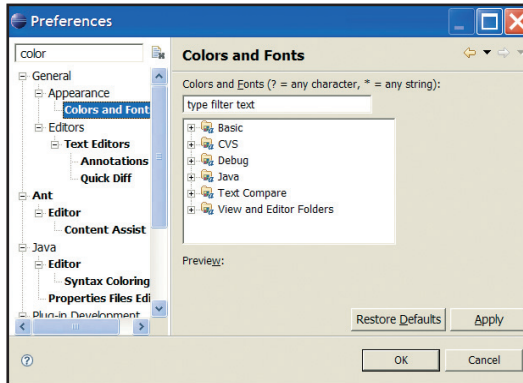
LE PROPRIETÀ DEL PROGETTO

Il nostro progetto è stato creato con le impostazioni di default, ma possiamo sempre decidere di cambiarle. Selezionate il progetto nel Package Explorer e scegliete Properties dal menu contestuale (o dal menu Project di Eclipse).
L'albero sulla sinistra divide le (molte) proprietà del progetto in categorie. Info mostra alcune informazioni generali, come il nome del progetto o il tipo di codifica del testo. Builders contiene la lista dei "costruttori" del programma. Il builder predefinito è il Java Builder, che fa il minimo indispensabile: mette nella cartella di output le classi compilate e tutto ciò che non è un file sorgente (se ad esempio provate ad aggiungere un file TXT alla directory src, il Java Builder lo copierà nell'output). Se il progetto richiede un processo di build più sofisticato, ad esempio per far girare dei test o fare il deploy remoto dell'applicazione, potete scrivere un file di Ant e usare l'Ant Builder di Eclipse al posto del Java Builder.

Java Build Path è una categoria importante, e contiene quattro tab. Source stabilisce dove devono essere il sorgente e le classi compilate. Di solito il sorgente è tutto nella stessa directory, ma potete usarne più d'una per separare i test dal codice di produzione, o il codice generato da quello scritto a mano, eccetera. Di Libraries parliamo nel box La biblioteca di Babele. Le altre due tab definiscono le dipendenze tra progetti. Ciascun progetto in un workspace può usare altri progetti (Projects) e può a sua volta essere usato, in tutto o in parte, da qualcun altro (Orders and Export).
Le altre categorie permettono di configurare centinaia di opzioni, tra cui la formattazione del sorgente e le impostazioni del compilatore. Molte tra queste impostazioni possono anche essere definite globalmente per l'intero workspace. Quindi in Eclipse esistono due livelli di configurazione: uno per il workspace e l'altro per i singoli progetti. Ciascun progetto eredita la gran parte dei propri parametri dal workspace, e può sovrascriverli se ha esigenze particolari. Nel prossimo passo troveremo una scusa per vedere come si cambiano le impostazioni del workspace.

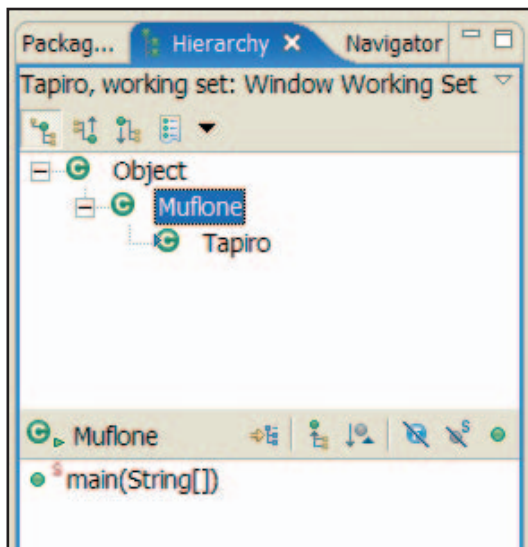
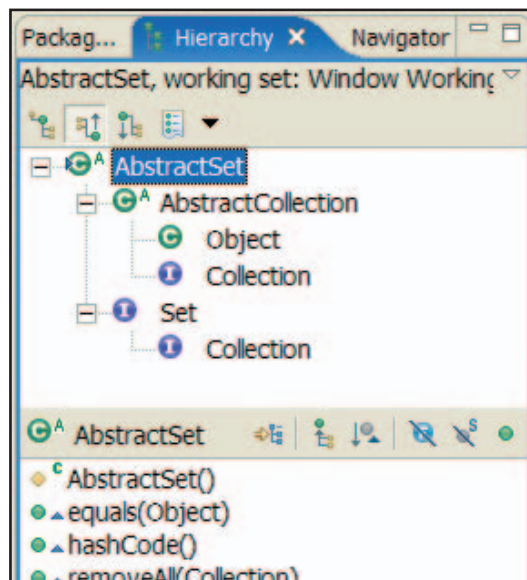
CERCARE LE INFORMAZIONI

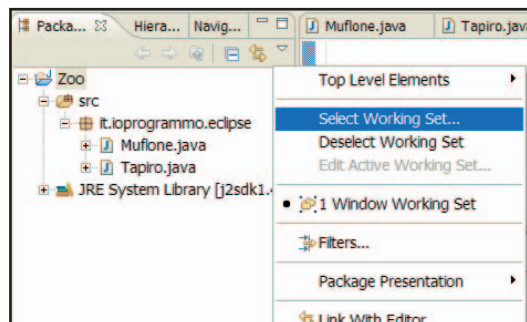
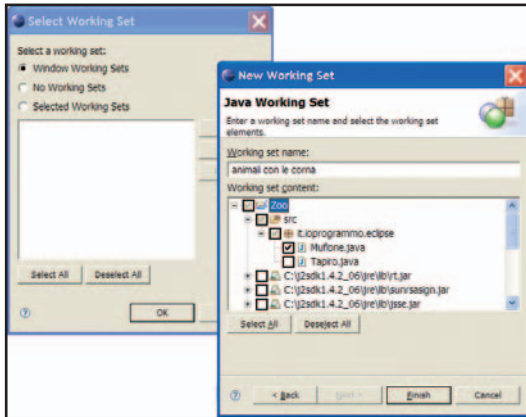
In un sistema complesso come Eclipse, ci sono sempre un sacco di informazioni. La gran parte dei progetti includono le API di Java, che hanno ormai superato le tremila classi, e probabilmente altre centinaia o migliaia di classi di libreria (vedi box: La biblioteca di Babele). Un grande progetto può avere centinaia di migliaia o anche

milioni di righe di codice. Persino la configurazione dello stesso Eclipse contiene centinaia di possibili parametri. Senza un sistema di ricerca efficiente, lavorare con tanta roba sarebbe un incubo.
Poniamo di voler cercare una classe o un'interfaccia. Il nostro progetto contiene una sola classe, ma include anche tutte le librerie di Java. Poniamo di aver bisogno della classe java.util.LinkedList, e facciamo un piccolo esperimento: è possibile trovarla e aprirla in meno di tre secondi? Premiamo Ctrl+Shift+T, e apparirà la finestra Open Type. Basta scrivere alcune lettere del nome della classe nella casella di testo, e la lista di tipi sottostante mostra subito la classe che ci serve, e che possiamo aprire con un doppio click. La lista delle classi è abbastanza intelligente per mostrare in cima le classi che

abbiamo già aperto di recente. Si possono anche usare “wildcards” (come il classico asterisco che significa “qualsiasi stringa”) e abbreviazioni (“LL” per “LinkedList”). Con un po' di pratica, si trova subito qualsiasi classe o interfaccia anche in un progetto enorme. E' vero, questa volta ci abbiamo messo più di tre secondi, ma con un po' di allineamento atletico potete arrivare a due secondi o poco più.
Cosa è successo, quando avete fatto doppio click sulla classe? Si è aperto il sorgente, oppure un mesto annuncio che dice che Eclipse non ha trovato il file .java? La cosa dipende dal fatto che la JVM di default del sistema includa o meno il file src.zip. Se questo file esiste, Eclipse è abbastanza furbo da andarci a guardare dentro per trovare i sorgenti delle API di Java. L'SDK di Java include i sorgenti, il Java Runtime no.
Ma, a proposito, come si fa a dire ad Eclipse quale JVM usare se ne abbiamo installata più di una? Proviamo a cercare questa opzione nella configurazione del programma, così potremo sperimentare la ricerca anche in quel campo (se siete stati sfortunati, avete già dovuto occuparvi di questa configurazione al Passo 1).
Dal menu Window scegliete Preferences. Si aprirà la finestra delle opzioni del workspace (ricordate che molte di queste possono essere “riconfigurate” per ciascun progetto).
Questa finestra contiene talmente tanta roba che ci vorrebbe molto tempo per trovare le opzioni più esoteriche, se non fosse per la combo in alto a sinistra. Provate a scriverci den-

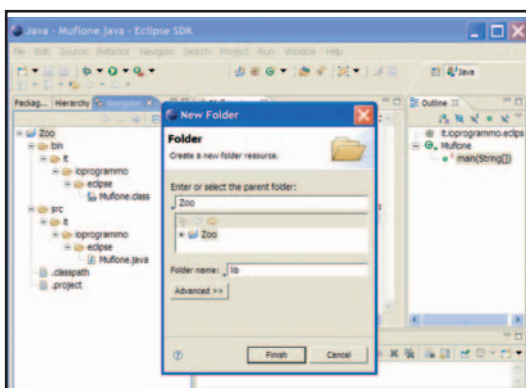
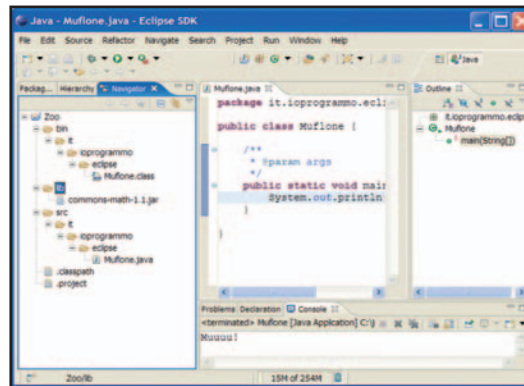
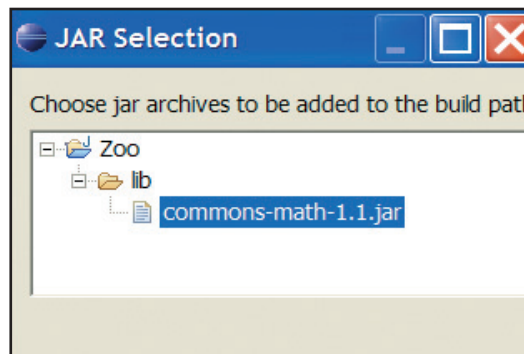
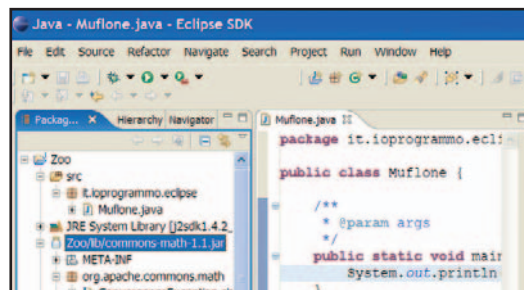
tro “jre”, ed ecco che magicamente l'albero delle opzioni si riduce a quello che ci serve. La funzionalità trova qualsiasi testo nella finestra, anche quello delle label. Provate a cercare nello stesso modo la configurazione della sintassi colorata, o quella dei warning del compilatore.
Il “quick search” dinamico è una bellissima cosa. Ma a volte fa comodo una ricerca più tradizionale, ad esempio quando si vuol cercare una particolare stringa in un file di sorgente. Il menu Search->Search mostra tutte le opzioni di ricerca che vi possono venire in mente, e probabilmente qualcuna in più. Anche in questo caso il sistema supporta le wildcard (e anche le regular expression, per chi vuole fare delle ricerche di testo sofisticate).

NAVIGARE LE CLASSI

Il browser delle classi è un componente essenziale degli IDE per i linguaggi object-oriented. In Eclipse il browser delle classi è la vista Hierarchy. Se non l'avete spostato, è sulla sinistra insieme al Package Explorer.
Per prima cosa ci serve una gerarchia di classi da esplorare. Selezionate il package it.ioprogrammo.eclipse nel Package Explorer e scegliete New->Class dal menu contestuale. Chiamate la classe Tapiro, ma aspettate a cliccare su Finish. Il campo Superclass contiene la superclasse di default java.lang.Object, mentre noi vogliamo che Tapiro sia una sottoclasse di Muflone (che il tapiro derivi dal muflone è un fatto ancora ignoto alla biologia, ma un mio imminente articolo sulla Wikipedia fugherà ogni dubbio circa questa verità dell'evoluzione). Cliccate sul tasto Browse accanto al campo Superclass per far

apparire una finestra di ricerca delle classi, e usatela nel solito modo per trovare Muflone. Ora potete creare la classe. Ovviamente avremmo potuto creare una classe che eredita da Object e poi aggiungere la derivazione a mano nel codice; ma in generale conviene usare gli strumenti automatici di Eclipse, che riservano molte piacevoli sorprese. Ad esempio, se Muflone fosse una classe che contiene metodi astratti, Eclipse aggiungerebbe automaticamente gli "stub" dei metodi nella nuova sottoclasse. Provare per credere.
Ora abbiamo una classe madre Muflone e una classe figlia Tapiro. Selezionate Tapiro nell'editor o nel Package Explorer e scegliete Open Type Hierarchy dal menu contestuale (lo shortcut da tastiera è F4 - conviene impararlo, perché lo si usa in continuazione). La vista Hierarchy passerà automaticamente in primo piano per mostrare l'albero genealogico di Tapiro. La parte inferiore della Hierarchy funziona quasi esattamente come la vista Outline. I tre piccoli tasti in cima alla vista permettono

di selezionare gli antenati della classe, i discendenti, o entrambi. Ora metteremo a dura prova il povero Eclipse. Selezionate Object, premete F4 e andate a farvi un panino. Al vostro ritorno vedrete l'intero albero di derivazione di Object – cioè tutto l'albero delle classi delle librerie Java (se usate un computer vecchiotto, potreste anche avere il tempo di finire il panino). È un buon momento per esplorare le API. Provate a selezionare la classe AbstractCollection e premere F4, e la vista si concentrerà su questa classe. Appariranno i discendenti di Abstract Collection, e anche la sua superclasse Object. Se volete vedere anche le interfacce implementate da AbstractCollection, premete il tasto Show the Supertype Hierarchy. Il punto di vista si inverte, e appare un albero dove la radice è AbstractCollection, e i discendenti sono tutti i suoi supertipi (la superclasse e le interfacce). Tornate alla vista dell'intero albero o a quello dei soli discendenti, ed esplorate i supertipi di AbstractSet per vedere una zona dove, tra classi e interfacce, la gerarchia diventa più complicata.

USARE I WORKING SET

In un progetto che contiene centinaia di classi, il Package Explorer può diventare affollato. Eclipse ha uno strumento fatto apposta per concentrarsi su "pezzi" di progetto: i working set. Un working set è un sottoinsieme dello spazio di lavoro. A destra della piccola toolbar privata del Package Explorer c'è una freccetta che punta verso il basso. Questa freccetta contiene un menu con un po' di operazioni aggiuntive, tra cui Select Working Set. Sceglietela, e apparirà la finestra dei working set. Premete su New per creare un nuovo working set. Scegliete Java (ci sono diversi tipi di working set) e poi Next. Date un nome al working set, come ad esempio "animali con le corna", e scegliete solo le classi

su cui volete lavorare (Muflone). Poi cliccate su Finish, selezionate il working set che avete appena creato e premete OK. Ora nel Package Explorer apparirà solo la classe Muflone. Tutto il resto è ancora nel progetto, naturalmente, ma non si vede. In questo caso non abbiamo avuto

grandi vantaggi – ma se vogliamo lavorare su un paio di package in un progetto che ne include dozzine, questa funzionalità diventa utile a "ripulire" le viste. Anche perché Eclipse ha già aggiunto il working set "animali con le corna" appare nel menu del Package Explorer, e selezionarlo è questione di secondi. Quando volete tornare a vedere tutto il progetto, selezionate il working set di default (Window Working Set).

LA BIBLIOTECA DI BABELE

Se c'è una cosa buona di Java, è che un sacco di cose sono già state fatte. Oltre alle sterminate (ma a volte curiosamente manchevoli) API del linguaggio, si trovano in giro librerie open source che fanno quasi tutto. Ciascuna di queste librerie dipende di solito da altre librerie, e alla fine qualsiasi progetto Java di medie dimensioni finisce per includere dozzine di file JAR. Vediamo come aggiungere questi JAR ad Eclipse.

1 Andate nel Navigator, selezionate la root del progetto e scegliete New->Folder dal menu contestuale per creare una nuova cartella per le librerie. Noi l'abbiamo chiamata lib.

2 Usando il drag&drop, copiate la libreria nella cartella lib. Nel nostro caso abbiamo deciso di usare commons-math-1.1.jar, una di quelle librerie di Apache che sono un po' come il prezzemolo.

3 Dal menu contestuale del progetto o dal menu Project scegliete Properties, e cercate la sezione Java Build Path. Cliccate su Add Jars e vedrete i file JAR contenuti nel progetto (è anche possibile aggiungere librerie che non risiedono nella cartella del progetto). Aggiungete commons-math-1.1.jar.

4 Andate nel Package Explorer e vedrete la libreria. Tutte le sue classi sono disponibili nel progetto, né più né meno delle classi delle API di Java, o di quelle scritte da noi.

Paolo Perrotta

SVILUPPARE CON DARWIN

IN NATURA LE SPECIE SI ADATTANO ALL'AMBIENTE CIRCOSTANTE IMPARANDO DALLE PROPRIE ESPERIENZA. PERCHÈ NON FARE LO STESSO CON IL SOFTWARE? PARLEREMO DI ALGORITMI EVOLUTIVI, OVVERO DI APPLICAZIONI CHE "MATURANO" NEL TEMPO

REQUISITI

Conoscenze richieste
-

Software
-

Impegno

Tempo di realizzazione

OLa programmazione genetica si colloca nel più vasto campo degli algoritmi evolutivi o calcolo evolutivo. Sotto la sigla algoritmi evolutivi si riconoscono una serie di tecniche : appunto la programmazione genetica, gli algoritmi genetici, le strategie evolutive, la programmazione evolutiva e i sistemi classificatori. Tutte si basano sulla biologia che descrive il meccanismo dell'evoluzione, siano esse particelle facenti parte di più ampie aggregazioni come insiemi di cellule o siano semplici popolazioni come formiche. Il processo evolutivo di un "individuo" è scandito da importanti stadi comuni a tutte le tecniche citate. Si tratta della *selezione*, *mutazione* e *riproduzione*; in linea con la teoria evolutiva darwiniana. Il processo di selezione dipende dalle *prestazioni* conosciute anche come *fitness* che gli individui garantiscono e che hanno sviluppato vivendo in quel determinato *ambiente*. Una popolazione si sviluppa in funzione di regole di selezione e operatori genetici, che esprimono la ricombinazione e la mutazione. La riproduzione si concentra su individui con elevate misure di fitness. Queste semplici regole, che sono molto vicine alla reale biologia che descrive l'evoluzione naturale, sono sufficienti a descrivere complesse realtà, e si sono dimostrate ottimi modelli di simulazione.
In pseudocodifica possiamo vedere come si può descrivere un generico algoritmo evolutivo. Si considera un sistema discreto per il quale il tempo evolve di singole unità. Ad ogni istante di tempo vengono ripetute le funzioni proprie del modello evolutivo.

```
// si parte fissando una variabile tempo
t=0
// si individua una popolazione iniziale (ad esempio in
                                              modo random)
initpopulation(P,t)
// Si valuta il fitness di tutti gli individui della
                                              popolazione
evaluate(P,t)
// ciclo per la valutazione continua della popolazione
// come criterio di terminazione si può considerare il
                                        tempo o il fitness
while not fatto do
{     //Incrementa il tempo
      t = t +1
      //Selezione della sottopopolazione per la
                                    produzione della prole
      Ps=selectparents(P,t)
      //ricombinazione dei geni della popolazione
                                              selezionata
      Ps=recombine(Ps,t)
      //Perturbazione stocastica della popolazione –
                                                mutazione
      Ps=mutate(Ps,t)
      //Valuta un nuovo fitness
      evaluate(Ps,t)
      //Selezione dei sopravvissuti dal processo
                                                di fitness
      P=survive(Ps,t)
}
```

La funzione *selectionparent* seleziona gli individui più promettenti in base alla precedente (*evaluate*) valutazione del fitness dei singoli. In particolare, si calcolano le misure di fitness e si ordinano in modo da poter selezionare un congruo numero di individui che assicurano una più "selezionata" popolazione, ovvero che abbiano migliori parametri in termini di fitness. Siamo molto vicini alla teoria ev olutiva, stiamo prendendo spunto dalla natura.

LA STORIA DEGLI ALGORITMI EVOLUTIVI

Le fondamenta storiche circa gli algoritmi evolutivi si devono a John Holland. Egli applicò i suoi studi ai sistemi biologi e artificiali. Un importante teorema che costituisce la base della teoria dimostra sotto determinate ipotesi che individui con alti valori di fitness tendono a crescere in modo esponenziale grazie al meccanismo dell'incrocio. Così si può raggiungere una soluzione ottimale.

ALGORITMI GENETICI

Un algoritmo genetico è un modello di apprendimento automatico – "learning machine" che deriva dal comportamento di alcuni dei meccanismi che descrivono l'evoluzione in natura. La popolazione di individui è qui rappresentata come cromosomi, in pratica un insieme di caratteri analoghi ai cromosomi base 4 presenti nel DNA.
Quindi gli individui nella popolazione simulano un processo di evoluzione. Uno dei campi in cui sono applicati gli algoritmi genetici è l'ottimizzazione, dove le stringhe di caratteri del cromosoma indicano il valore di differenti parametri di ottimizzazione. Buoni risultati si sono ottenuti anche in problemi classici, in particolare laddove la funzione obiettivo era discontinua o fortemente non lineare. In sostanza gli algoritmi genetici sono un sottoinsieme degli evolutivi, ma per essi l'individuo è rappresentato come set di cromosomi. Entriamo nei dettagli di funzionamento di un algoritmo genetico. La struttura intorno alla quale ruota l'intero metodo è una stringa detta gene che descrive la sequenza di cromosomi. Una funzione restituirà una sorta di valore del gene, è il già citato fitness. Come descritto in precedenza l'obiettivo dell'algoritmo è modificare nel tempo la popolazione dei geni in modo da migliorarla di continuo.
Un elemento fondamentale di differenziazione rispetto al calcolo evolutivo è la fase di riproduzione che qui avviene come combinazione delle due fasi di mutazione dei genitori e di conseguente incrocio dei geni (conosciuto come crossover).
Il crossover che deriva dalla sintesi delle due parole "crossing over" è il modo in cui vengono combinati tra loro i geni dei due genitori per la produzione di geni figlio. Esistono molti modi per simulare tale "mescolamento". Modelli sofisticati sono in grado di tenere conto di molte delle informazioni contenute nella lunga catena cromosomica. Ad ogni modo, per intenderci e capire come può essere simulata tale fase, possiamo esaminare il basilare metodo di "single point crossover".
Con tale tecnica le due catene vengono tagliate alla stessa lunghezza formando cosi entrambe due tronchi: una testa ed una coda. Si tratta a questo punto di produrre il nuovo gene con i pezzi risultanti dal troncamento. Quindi con la testa di uno dei due geni e la coda dell'altro. La mutazione, come ci ricordano gli studi di scienze, consiste nella modificazione di alcuni dei geni della sequenza. L'evoluzione avviene appunto andando a selezionare le mutazioni che producono un maggiore fitness. Nell'implementare eventuali algoritmi bisogna quindi descrivere i geni come array e applicare ad essi le varie fasi descritte.

DALLA PROGRAMMAZIONE EVOLUTIVA AI SISTEMI CLASSIFICATORI

Diamo cenno delle altre tecniche di calcolo evolutivo. La programmazione evolutiva introdotta da Lawrence J. Fogel, è una strategia di ottimizzazione stocastica simile agli algoritmi genetici. Viene però qui enfatizzato, anziché il comportamento della fase riproduttiva tra genitori e figli, l'analisi della crescita della popolazione in funzione degli operatori genetici applicati. La strategia evolutiva è molto simile alla precedente, punta comunque l'interesse al raggiungimento degli obiettivi in termini di evoluzione della popolazione osservata come macro processo, ma questa volta come conseguenza di mutazioni che avvengono in modo casuale e delle conseguenti selezioni. Tale teoria è stata introdotta da tre studenti dell'università di Berlino, oggi professori, I. Rechenberg, a H.P. Schwefel e P. Bienert. Il nome sistema classificatore è stato il primo nome dato dal capostipite J. Holland alla teoria. Oggi si intendono modelli di apprendimento che non hanno componenti evolutive. Quindi un sistema classificatore partendo da zero genera casualmente una popolazione, successivamente il sistema impara a mutare per induzione. Cosi la popolazione si sviluppa e "impara" a reagire ad eventuali cambiamenti dell'ambiente. Un sostanziale contributo alla teoria è stato dato dall'italiano Marco Dorigo.

PROGRAMMAZIONE GENETICA

La programmazione genetica o GP è stata sviluppata fondamentalmente da J. Koza. Si tratta di un metodo per la generazione automatica di programmi capaci di implementare un predeterminato compito. Si fonda, come gli altri metodi, sulla selezione naturale descritta dalla teoria evolutiva di Darwin, in modo da realizzare programmi che siano sempre più efficienti. Così, una sua applicazione altera il codice sorgente del programma; insomma è un algoritmo evolutivo per il quale la popolazione è il programma e il gene, ossia la stringa di lunghezza costante, diventa un programma rappresentato da una struttura ad albero. In questo ambito non si presenta alcuna mutazione ma solo riproduzioni.

Un modo per rappresentare un programma genetico fa uso di alberi, come mostrato in figura 1 dall'esempio che lo stesso Koza ha proposto.

L'equivalente espressione Lisp della funzione è: (+ 1 2 (IF (> TIME 10) 3 4)).
Va sottolineata la presenza di una chiara struttura gerarchica che ha sostituito la piatta stringa gene, presente negli algoritmi genetici. In figura 1 per chiarezza si è riportato un semplice esempio che moltissimi linguaggi di programmazione ad alto livello sono già in grado di implementare. Si parte da un albero iniziale che verrà manipolato attraverso gli operatori genetici, come il crossover e la riproduzione, ad ogni iterazione del ciclo fino ad arrivare ad un programma completo e funzionante. Vi è una distinzione tra insiemi da trattare. Un primo insieme

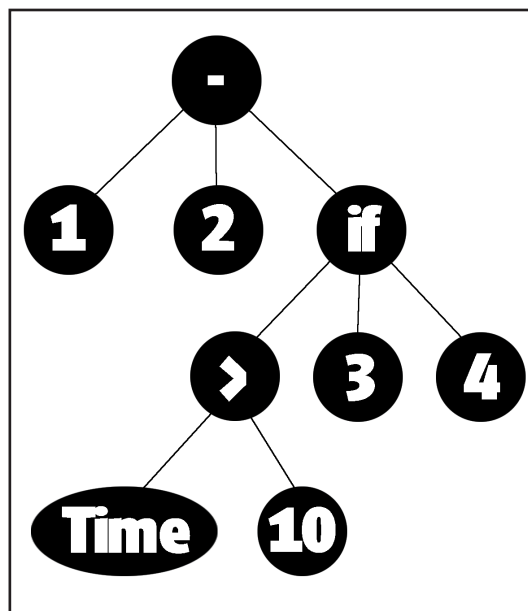



Fig. 1: "Una semplice struttura ad albero che esprime una funzione"

T contiene i terminali, un secondo F le funzioni. {1, 2, 3, 4, 10, TIME} è sottoinsieme di T; {+, >, IF} è sottoinsieme di F. I terminali contengono quindi variabili o costanti. Per costruire l'albero si opera una intersezione tra i due insiemi facendo accortezza a considerare come foglie gli elementi appartenenti a T. I quattro operatori genetici maggiormente usati sono: crossover, riproduzione, mutazione e inversione o permutazione. L'effetto del crossover è chiaramente descritto nella figura 2.
Si scelgono due sottoalberi dai due genitori, estraendo casualmente due nodi. I due alberi prodotti come figli si ottengono unendo i sottoalberi prima estratti. Tale procedimento attua anche la riproduzione visto che si generano due nuovi programmi. Va sottolineata la diversità che viene così garantita. L'eventuale riproduzione "incestuosa"di uno stesso individuo con se stesso, ossia due identici alberi posti a crossover, produce generalmente diversi figli. Ma, mentre le operazioni di riproduzione darwiniana portano alla convergenza; nella programmazione genetica le operazioni di crossover esercitano un contributo contrario alla convergenza. La mutazione, che dovrebbe garantire diversità alla popolazione nella programmazione genetica, così come proposta da Koza, si attua grazie all'applicazione delle operazioni di crossover. Così, la mutazione non è altro che una variazione sulle operazioni di crossover. L'inversione, ossia la permutazione di alcuni "pezzi" anche se proposta da Koza non è mai stata effettivamente utilizzata. Nella riproduzione si ha il compito di selezionare tra i figli prodotti. Per farlo si usano le funzioni che restituiscono il fitness, ossia una valutazione della "bontà" del programma prodotto. In generale, anche per gli algoritmi genetici, la valutazione del fitness è un argomento con molti punti interrogativi.
Esistono principalmente due modi per attuare tale selezione: scegliere l'individuo che ha la più alta misura in funzione della riproduzione, il che potrebbe essere tradotto in "solo i più forti sopravvivono"; oppure scegliere l'individuo con un più ricco contenuto genetico in modo che sia favorita la diversità. Ad ogni modo la scelta della funzione del fitness è soggettiva e dipende anche dall'ambito di applicazione. Ad esempio si potrebbe includere la profondità dell'albero come potenziale qualità che si desidera controllare e sviluppare quindi una funzione di fitness che ne tenga conto.

PRODUZIONI DELLA PROGRAMMAZIONE GENETICA

Inizialmente, come lo stesso Koza fece, si preferiva sviluppare programmi per GP in linguaggi come il lisp organizzato in strutture ad albero. Successivamente, si sono utilizzati anche i linguaggi imperativi e conseguenti rappresentazioni lineari. Alcuni tra i più importanti programmi di GP sono cosi prodotti. Esempi sono: il software commerciale "Discipulus" utilizzato in molti campi da quello commerciale a quello medico; e "MicroGP" che usa la programmazione genetica lineare per generare programmi in codice assembly.

IMPLEMENTAZIONE

Il linguaggio naturale per l'implementazione della programmazione genetica è il Lisp, per la sua struttura ad albero. Anche altri linguaggi possono essere usati con successo. Attenendo-

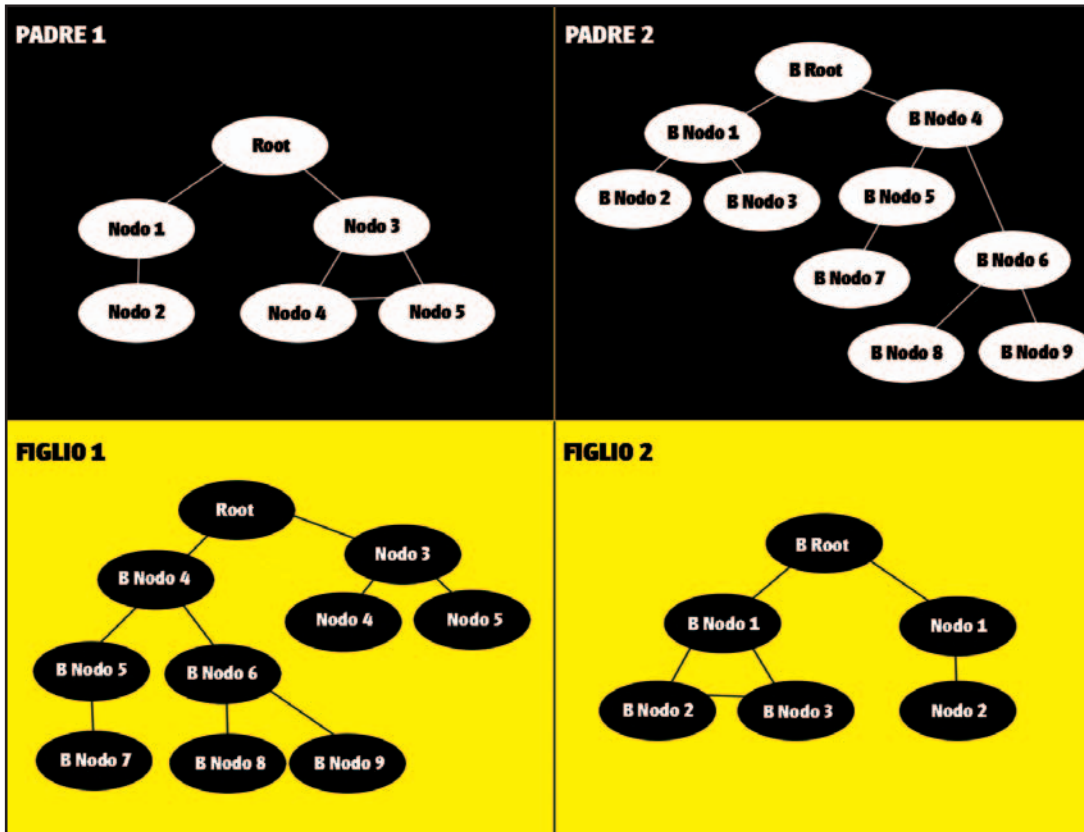



Fig. 2: **"Effetto dell'operazione di crossover. Dai due genitori vengono prodotti due figli"**

ci alle nostre abitudini sviluppiamo usando il C++. Sviluppare l'intero progetto non è affatto una passeggiata, appelliamoci quindi alla metodologia top down. Dividiamo il lavoro in tre moduli:

1. Definizione degli operatori disponibili dagli alberi del programma;
2. Definizione delle classi e delle operazioni che compongono gli alberi del programma;
3. Sviluppo del meccanismo per la produzione della popolazione di alberi di programmi; si includono in questa fase le funzioni di riproduzione

Il primo modulo dipende dal tipo di applicazione che si sta sviluppando e va quindi progettato e sviluppato ogni qual volta si intende produrre una nuova applicazione. Gli altri due moduli non necessitano di cambiamenti.
Progettiamo la struttura dati. Si implementano due tipi di operatori, uno binario e l'altro unario. Ecco i tipi che li descrivono.

```
typedef struct Binary_Func {
    Val (*eval)(Val,Val);           // la funzione
                                          puntatore
    char *sign;                     // operatore infisso
                                               sign
};
```

```
typedef struct Unary_Func {
    Val (*eval)(Val);               // la funzione
                                          puntatore
    char *sign;                     // operatore prefisso
                                               sign
};
```

val può cambiare in funzione del problema da risolvere. **sign* è una stringa usata per poter visualizzare gli alberi di programma in modo leggibile. La stringa è tra i due operatori per le funzioni binarie, mentre è prefissa per gli operatori unari. Sviluppiamo adesso a titolo di esempio alcune operazioni elementari come la somma e il valore assoluto. Ecco il codice

```
// funzione per la somma di due valori

Val add(Val a, Val b)
{
    return (a+b);
}
// il carattere sign per la somma
char add_sign[] = " + ";
// la rappresentazione binaria della funzione
Binary_Func Add = {add, add_sign};

// funzione valore assoluto
Val abs(Val a)
```

```
{
    if (a < (Val)0)
      return (-a);
    else
      return a;
}
char abs_sign[] = "ABS";
Unary_Func Abs = {abs, abs_sign};
```

La struttura delle variabili contiene un puntatore al tipo *val*, e una stringa utili alla stampa. Le variabili vere e proprie sono definite nel modulo applicativo.

```
// struttura delle variabili (usata per variabili
// terminali).
// si consente così la stampa
typedef struct Variable {

    Val *var;                    // puntatore al valore
                                // della variabile
    char *name;                  // nome della variabile
};
```

Gli alberi del programma sono costituiti da nodi. Essi fanno riferimento ad un nodo radice (root). Si definisce *node* come classe virtuale. Ecco il codice:

```
// base class for the 4 types of nodes

class Node {
  protected:
    Node *parent;                // nodo padre
    int n_children;              // numero di nodi figlio
    int n_valid;                 // flag di validità

  public:
    Node(void);                  // construttore
    ~Node(void);                 // distruttore: resetterà
                                          il puntatore di
                                 // questo nodo
                                                del padre

    int type;                    // indica il tipo di nodo
    void invalidate_count(void); // chiamato quando
                                 il numero di figli di questo
                                              // nodo è
                                                cambiato
    virtual Val value(void) = 0; // Per assegnare valori
                                              a questo nodo
     virtual int count(void) = 0; // restituisce il
                                              numero di figli
    virtual Node *find(int) = 0; // usata per restituire
                                         un particolare nodo
    virtual char *print(void) = 0; // visualizza
              l'espressione sottoalbero come una stringa
    // resettta il nodo padre
    void set_parent(Node *n) {parent = n;};
    // accede al nodo padre
    Node *get_parent(void) {return parent;};
};
```

Anche se il codice è commentato esaminiamo alcuni aspetti. **parent* è il puntatore al nodo genitore del nodo puntatore corrente (this). I due membri protetti *n_children* e *n _valid* indacono rispettivamente il numero di figli (ossia un sottoalbero) e un flag che innesca, quando necessario, una ricomputazione. Entrambi non possono essere qui usati, ma solo per classi derivate. La funzione *invalidate_count* viene richiamata quando cambiano il numero di figli, solitamente in seguito ad operazioni di crossover o riproduzione, tale informazione è resa disponibile da *n_valid*. Con *value()* si valuta un nodo a partire dai suoi figli. Per valutare l'intero albero è necessario sottoporre la funzione alla radice. Il membro *find()* trova un figlio nell'albero mentre *print()* visualizza ricorsivamente il sottoalbero specificato.

CONCLUSIONI

La realizzazione del progetto prevede altri importanti passi che ci riserviamo di esaminare nel prossimo appuntamento. Si tratta di sviluppare della sottoclassi di nodo per l'implementazioni di funzioni non terminali, quindi le operazioni unarie e binarie.
Poi va costruita l'importante classe popolazione che contiene le funzionalità necessarie per creare un gruppo di alberi, assegnare fitness, selezionare i membri per la riproduzione e creare nuove generazioni usando il crossover. Insomma ci sono ancora importanti passi per concludere il progetto che sveleranno nuovi segreti e apriranno nuove interessanti finestre sulla programmazione genetica. Alla fine saremo in grado di creare programmi autoapprendenti che si basano sulle teorie evolutive

Fabio Grimaldi

ARTICOLI CORRELATI

Nel corso degli anni ho affrontato una serie di argomenti che gravitano intorno allo stimolante mondo della programmazione genetica. Con la firma Fabio Grimaldi sono stati sviluppati nella sezione di soluzione i seguenti articoli: Il gioco della vita - n. 34;Lunga vita al serpente - n.41; Il serpente ha ancora fame - n.42; Automi cellulari unidimensionali - n. 53; Modelli di contagio con automi cellulari - n.54; Un automa cellulare doppio - n.55.